TOSHIBA

TX03 ペリフェラルドライバ使用例 (TMPM384)

第一版

2017 年 9 月

東芝デバイス＆ストレージ株式会社

CMDR-M384UE-01J

本製品取り扱い上のお願い

- ソフトウエア使用権許諾契約書の同意無しに使用しないで下さい。

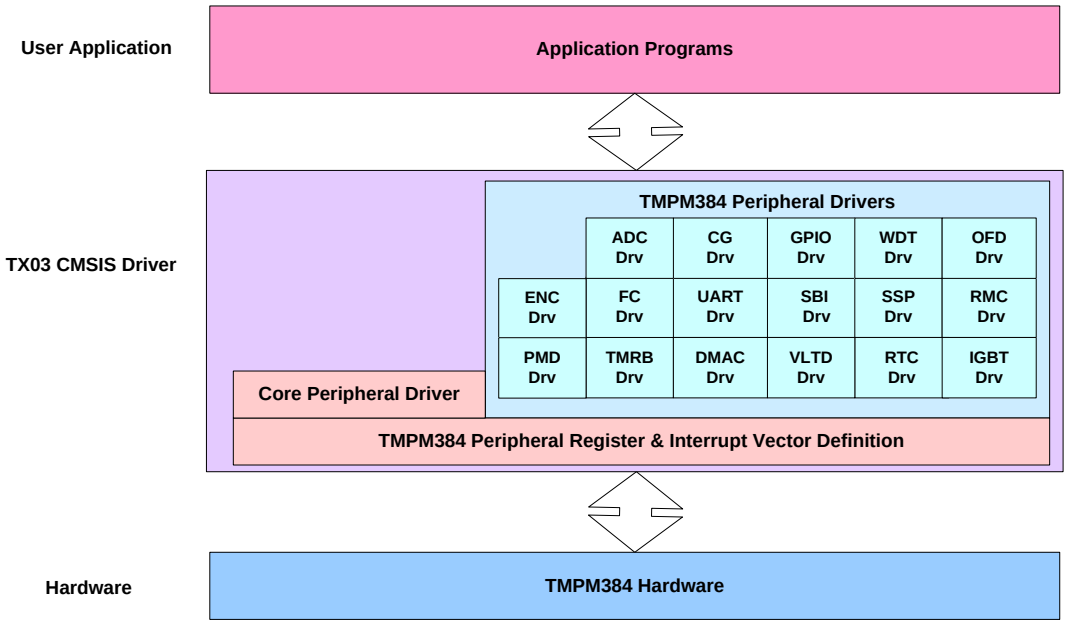
目次

1 はしがき

本アプリケーションノートのサンプルプログラムは、東芝製マイコンTMPM384用です。各サンプルプログラムは、主なMCU内蔵機能を単独で実行するように出来ています。サンプルプログラム内の一部を取り出して再利用することで、希望する機能を動作させることができます。

2 概要

TX03ペリフェラルドライバを下記のように使用します。



3 使用する機能

機能	チャネル	使用／未使用
クロック／モード制御 (CG)	クロックギア	使用: CG サンプル
	PLL	PLL on
低消費電力モード	-	使用: SLEEP モード
SysTick	-	未使用
DMAC	DMA1	使用: DMAC サンプル
ウォッチドッグタイマ (WDT)	-	使用: WDT サンプル
外部割り込み (INT)	INT0	使用: CG サンプル
シリアルチャネル(SIO/UART)	SIO1	使用: DMAC _UART サンプル
同期式シリアルインタフェース(SSP)	SSP0	使用: SSP 転送サンプル(Tx)
	SSP1	使用: SSP 転送サンプル(Rx)
シリアルバスインタフェース(SBI)	SBI0	使用: マスタ(Tx)

機能	チャネル	使用／未使用
	SBI1	使用: スレーブ(Rx)
リモコン判定機能 (RMC)	RMC0	使用: RMC 信号受信
16 ビットタイマ(TMRB)	TMRB0	使用: TMRB サンプル: 汎用タイマ
	TMRB6	使用: TMRB サンプル: PPG 出力
16 ビット多目的タイマ	MPT0	使用: IGBT サンプル: PPG 出力
	MPT1	使用: PMD サンプル: 位相出力
RTC	-	使用: RTC サンプル
周波数検知回路 (OFD)	-	使用: OFD サンプル
電圧検出回路(VLTD)	-	使用: VLTD サンプル
エンコーダ入力回路(ENC)	ENC0	使用: ENC サンプル
12 ビット/10 ビット A/D コンバータ	AIN8	使用: ADC データリード
Flash	-	使用: FC サンプル

4 端子用途

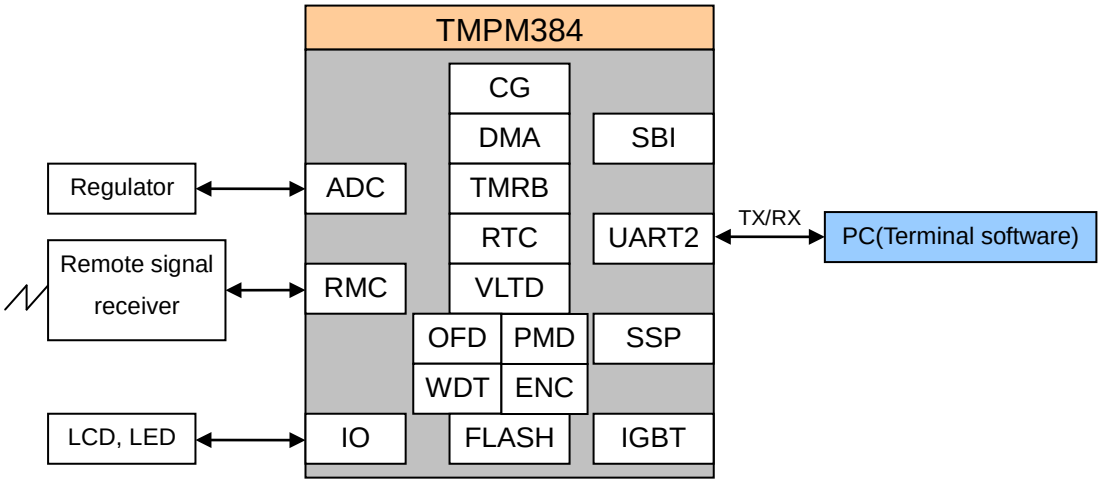
本サンプルプログラムは、TMPM384評価ボードを用いてテストされています。以下に、端子用途を説明します。

Pin No.	端子名	用途
92	PH0/AIN0/INT0	SW0/4
93	PH1/AIN1/INT1	SW1/5
94	PH2/AIN2	SW2/6
95	PH3/AIN3	SW3/7
33	PT0/TB8IN	LED0
34	PT1/TB8OUT	LED1
35	PT2/TB9IN	LED2
36	PT3/TB9OUT	LED3
20	PC1/XO0/SP0DI/SCL0/SI0	マスタ(SBI0) SCL0
21	PC0/UO0/SP0DO/SDA0/SO0	マスタ(SBI0) SDA0
29	PG0/UO1/SDA1/SO1	スレーブ(SBI1) SDA1
30	PG1/XO1/SCL1/SI1	スレーブ(SBI1) SCL1
14	PC7, MT0IN	IGBT0トリガ入力
15	PC6, GEMG0	IGBT0 EMG入力
17	PC4, MTOUT00	IGBT0 PPG出力0
16	PC5, MT0OUT10	IGBT0 PPG出力1
52	PA5,TB6OUT	TMRB6 PPG出力
50	PA3,RXIN	RMC受信
18	PC3	SSP SP0FSS
19	PC2	SSP SP0CLK
20	PC1	SSP SP0DI
21	PC0	SSP SP0DO
52	PA5	UART1 TX
65	PN0	SSP SP1DO
66	PN1	SSP SP1DI
67	PN2	SSP SP1CLK
81	PN3	SSP SP1FSS
100	PI0/AIN8	ADC AIN8
12	PD0/ENCA0/TB5IN/INTC	ENCA0
11	PD1/ENCB0/TB5OUT	ENCB0
116	DVSS	GND pin
117	DVDD5	+5V
29	PG0 / UO1 / SDA1/SO1	PMD UO1 output
30	PG1 / XO1 / SCL1/SI1	PMD XO1 output

Pin No.	端子名	用途
31	PG2 / VO1 / SCK1	PMD VO1 output
32	PG3 / YO1	PMD YO1 output
43	PG4 / WO1 / MTOUT01 / MTTB1OUT	PMD WO1 output
44	PG5 / ZO1 / MTOUT11 MTTB1IN	PMD ZO1 output
45	PG6 / EMG1 / GEMG1	PMD EMG1 input

5 開発環境

下記に開発環境の構成を示します:



1. ハードウェア:
   TMPM384 評価ボード+TMPM384FDFG

2. 開発ツール:
   - IAR 社:
     1) J-Link: IAR 社製 J-Link-ARM 7.0 / 8.0
     2) IDE:IAR 社製 Embedded workbench 6.40.2
   - KEIL 社:
     1) u-Link: Realview ULINK2
     2) IDE: KEIL uVision MDK 4.54

補足: ENC/PMD サンプル動作環境は次の通りです。

- IAR:
  1) J-Link: IAR J-Link-ARM 7.0 / J-Link 6.0
  2) IDE: IAR Embedded workbench 6.50.1 version
- KEIL:
  1) IDE: KEIL uVision 4.60

6 機能説明

6-1 動作モード選択

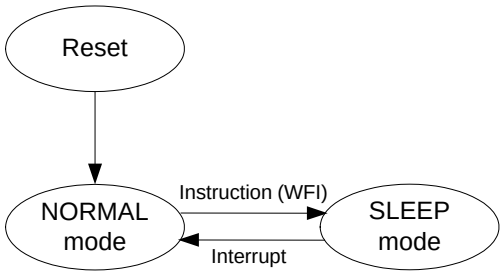
TMPM384 には 5 つの動作モードがあります: NORMAL, SLOW, IDLE, SLEEP, STOP モード
IDLE, SLEEP, STOP モードは低消費電力モードです。
低消費電力モードに移行するには、システム制御レジスタ STBYCR0<STBY2:0>にて IDLE, SLEEP, STOP モードのいずれかを設定し、WFI (Wait For Interrupt)命令を実行します。

- **NORMAL モード:**

CPU コアおよび周辺ハードウェアを高速クロック(fosc1 または fosc2)で動作させるモードです。
リセット解除後は、fosc2(内蔵高速クロック)による NORMAL モードになります。
補足: サンプルプログラムは、SLEEP モードのみ使用します。

- **SLEEP モード:**

SLEEP モードでは、低速発振(fs)、RTC、RMC が動作します。SLEEP モードが解除されると SLEEP モードへ移行する直前の動作モードへ復帰し、動作を開始します。
RTC 割り込み、RMC 割り込み、外部割込みで SLEEP モードを解除できます。
補足: SLEEP モードのサンプルプログラムは、CG の例に含まれています。



6-2 ADC

この例は、*AIN8*(pin100)に接続された VR2 の値を変更するサンプルプログラムです。
測定された電圧値により、評価ボード上の 4 つの LED の点滅間隔を変更します。
電圧値が高いほど点滅間隔が短くなります。

サンプルプログラムのシーケンス:

1. ADC クロックを設定します。

2. AD チャネルとスタートトリガを選択します。
3. 低電圧モードを禁止します。
4. ADC 回路を許可します。
5. AD 変換を開始します。
6. AD 変換終了後、変換結果を取得し、取得結果によって LED 点滅間隔を調整します。
7. 手順 6 を繰り返します。

6-3 CG

6-3-1 パワーモード変更

この例は、CPU の動作モードを変更するサンプルプログラムです。NORMAL と SLEEP の 2 モードのみサポートします。 パワーモードを切り替えるにはキーを押してください。

現在のモード	アクション (Key)	動作	LED 表示
NORMAL	SW1 を押す	NORMAL→ STOP1	全 LED 消灯
SLEEP	SW1 を戻す	STOP1 →NORMAL	全 LED 点灯

6-4 DMAC

6-4-1 メモリから周辺回路

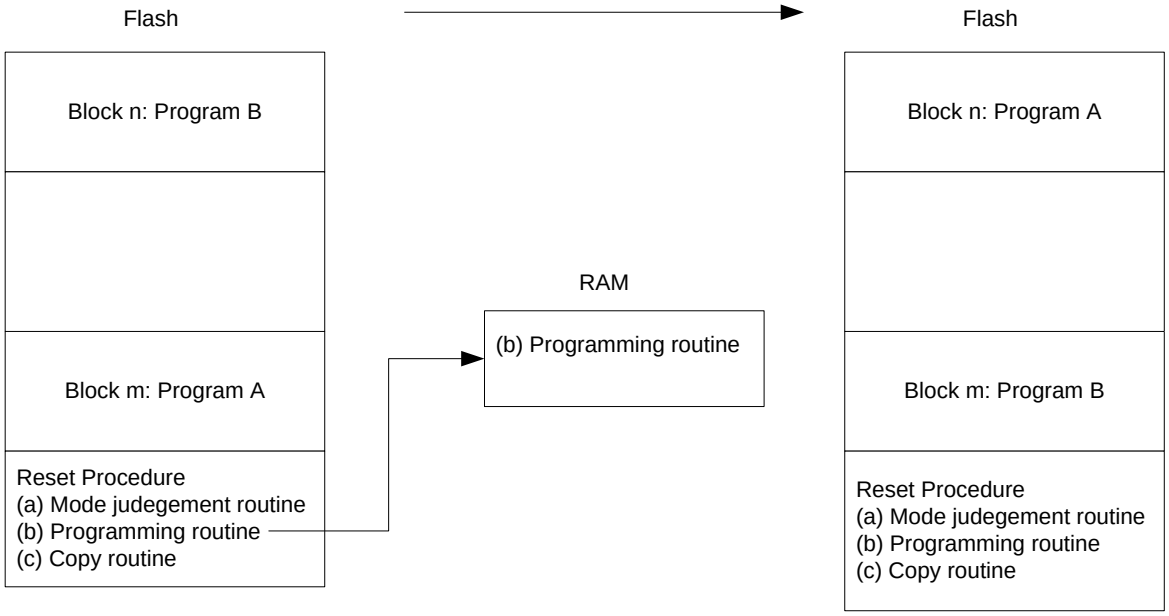
この例は、DMA が DST_Buffer[]にデータを転送し、TransferResult 変数と比較するサンプルプログラムです。

6-5 FLASH

6-5-1 Flash スワップ

この例は、Flash のドライバを使用して内蔵フラッシュメモリの消去及び再書き込みを行うサンプルプログラムです。
このプログラムは、シングルチップモードのノーマルモードとユーザーブートモードで実行します。
ーリセット動作: モード判定、書き込みルーチン(フラッシュ API)、転送ルーチンで構成されます。
・モード判定ルーチン: ユーザブートモードまたはノーマルモードの判定を行います。
・転送ルーチン: 書き込みルーチンを Flash から RAM へ転送します。
・書き込みルーチン: RAM 上で動作し、フラッシュ上のプログラム A とプログラム B のコードを入れ替えます。

ープログラム A/B (リセット動作)は事前にフラッシュメモリに書き込まれています。
ープログラム A/B (A: LED 0 の点灯とLCD に"DEMO A"を表示します。B: LED 1 の点灯とLCD に"DEMO B"を表示します。)
初期状態は、プログラム A が最初に動作します。
ーSW0 キーはリセット動作のモード判定に使います。
SW0 キーが押された場合 → ユーザブートモードです。
SW0 キーが押されていない場合 → Normal モードです。



プログラムシーケンス

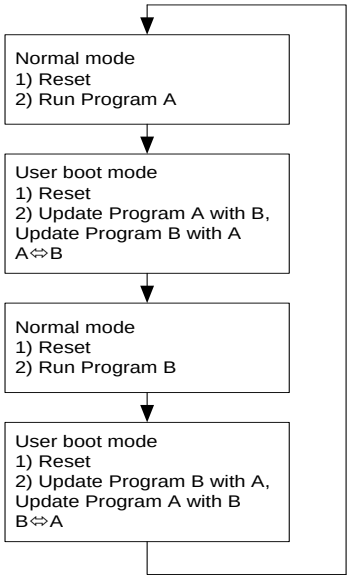
(1) 電源投入
SW0 を OFF した状態で電源投入またはリセット端子を ON します。
まず Flash 内に書き込まれているプログラム A を実行します。プログラム A は、LED0 の点滅と LCD に“DEMO A”を表示します。

(2) SW0 を ON しながら RESET ボタンを ON します。その後、LCD に“User Boot Mode” が表示されるまでに SW0 を OFF します。
リセット処理の中で転送ルーチンが書き換えルーチンを RAM へ転送します。
その後、Flash 内のプログラム A とプログラム B を入れ替えるコードを実行します。
この間、LCD は、以下のような表示を行います。
“RAM transferring ……”, “FLASH swapping ……” and “Finished Restart ……”.

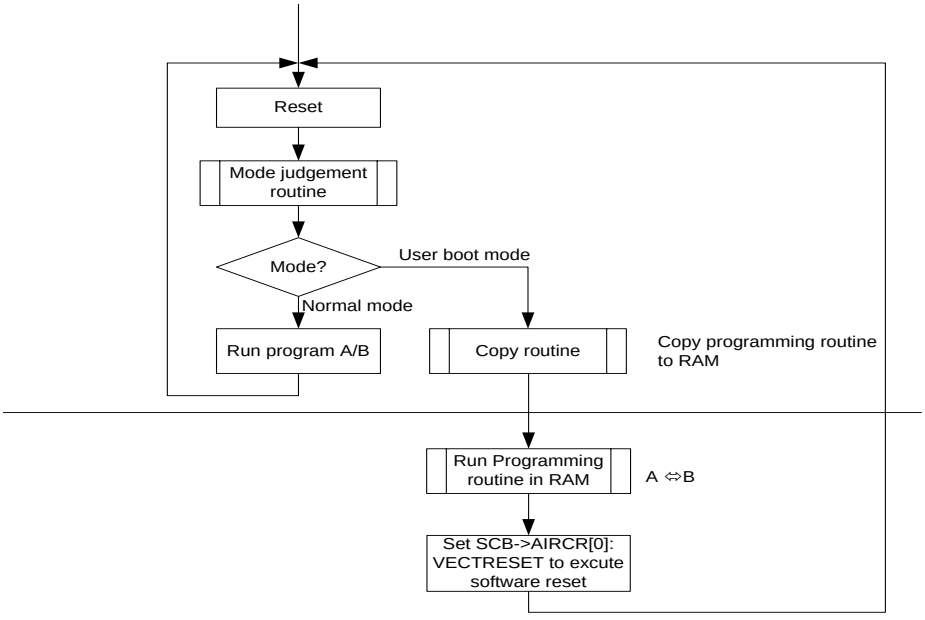
(3) Flash 内のプログラム A とB が入れ替わり、LCD に “Finished Restart ……”を表示すると、プログラムは自動的にソフトウェアリセットします。
その後、Flash 内に書き込まれたプログラム B が実行を始めます。
LED1 を点灯し、LCD に“DEMO B”を表示します。

(4) 再度 SW0 を ON しながら RESET ボタンを ON します。その後、LCD に“User Boot Mode” が表示されるまでに SW0 を OFF すると、手順 2 を繰り返します。

(5) LCD に“Finished Restart ……”が表示されると、プログラムは自動的にソフトウェアリセットします。
その後、Flash 内に書き込まれたプログラム A が実行を始めます。
LED0 を点灯し、LCD に“DEMO A”を表示します。



・フローチャート



サンプルプログラムがユーザブートモードに移行すると、LCD に以下を表示します。

User Boot Mode

(リセット処理)

RAM 転送か Flash 書き込みを行っている間に LCD は現在の状態を表示します。

以下は LCD 表示の例です。

RAM transferring ……	FLASH swapping ……	Finished Restart ……
(転送処理)	(プログラム処理)	(プログラム処理)

6-6 GPIO

この例は、GPIO を LED に設定し、LED の点灯/消灯を行うサンプルプログラムです。

6-7 OFD

この例は、クロックが OFD に設定された周波数範囲を超えると、OFD リセットを行い、OFD リセットフラグをセットするサンプルプログラムです。
フラグがセットされると、OFD は無効化され、LED は点灯します。

6-8 IGBT

6-8-1 シングル PPG 波形出力

この例は、IGBT や EMG を使用して PPG 出力を制御するための外部トリガの使い方のサンプルプログラムです。MPT チャネル 0 (IGBT0) を使用し、PPG 波形を MTOUT00 から出力します。

端子割り当て:

機能	端子名	Pin No.
トリガ入力	MT0IN / PC7	14
PPG 出力	MTOUT00 / PC4	17
EMG 入力	GEMG0 / PC6	15

シーケンス:

1. GEMG0 端子を外部プルアップするためにジャンパを使って DVDD と接続します。
2. MTOUT00 端子をオシロスコープに接続します。
3. 電源を投入します。
   LED0 が点灯し、MTOUT00 端子から何も波形が出力されていないことを確認します。.
4. 立ち下りエッジを入れるため MT0IN 端子を GND に接続します。

5. この立ち下がりエッジで IGBT タイマをスタートし、MTOUT00 端子からデューティ 50%の PPG 波形を出力します。
6. GEMG0 端子を GND と接続すると PPG 波形出力を停止し、LED1 を点灯します。
7. GEMG0 端子を DVDD と接続すると、EMG 状態を停止し、LED1 を消灯します。再度 PPG 波形を出力するには、手順 4 を繰り返します。

6-8-2 ダブル PPG 波形出力

この例は、IGBT や EMG を使用して 2 つの PPG 出力を制御するためのコマンドの開始方法のサンプルプログラムです。MPT チャネル 0 (IGBT0) を使用し、2 つの PPG 波形を MTOUT00 と MTOUT10 から出力します。

端子割り当て:

Function	Pin Name	Pin No.
PPG 出力 0	MTOUT00 / PC4	17
PPG 出力 1	MTOUT10 / PC5	16
EMG 入力	GEMG0 / PC6	15

シーケンス:

1. GEMG0 端子を外部プルアップするためにジャンパを使って DVDD と接続します。
2. MTOUT00 端子と MTOUT10 端子をオシロスコープに接続します。
3. SW7 を OFF にします
4. 電源を投入します。
   LED0 が点灯し、MTOUT00 端子と MTOUT10 端子から何も波形が出力されていないことを確認します。
5. SW7 を ON します。
6. IGBT タイマをスタートし、MTOUT00 端子と MTOUT10 端子からディーティ 40%(High レベル: 20μs)で周期 50μs の PPG 波形を出力します。また、MTOUT10 は MTOUT00 から 25μs ディレイしています。
7. SW7 を OFF すると、PPG 波形出力を停止します。
8. GEMG0 端子を GND と接続すると LED1 を点灯します。
9. GEMG0 端子を DVDD と接続すると、EMG 状態を停止し、LED1 を消灯します。再度 PPG 波形を出力するには、手順 5 を繰り返します。

6-9 RMC

6-9-1 RMC 受信

この例は、リモコン信号データを受信し、デコードするサンプルプログラムです。
またデコードしたデータをデバッガ上のターミナルウィンドウに表示します。
カスタムコードまたはアドレスコードとコマンドコードを Hex フォーマットで表示します。

表示フォーマット:　TMPM384 RMC Demo
RMC_1: XX XX
RMC_1: YY YY

フォーマット	ターミナルウィンドウの表示
1	RMC_1:
2	RMC_2:
3	RMC_3:
4	RMC_4:
5	RMC_5:

6-10　RTC

この例は、RTC を使用して LCD に日時を表示するサンプルプログラムです。
初期設定では 2010-10-22, 時刻の設定が 12:50:55, 24 時間表示です。
更新間隔: 1 秒
LCD 表示:

2010/10/22 12:50:55

6-11　SBI

この例は、I2C バスの 1 本を I2C マスタとして、もう片方を I2C スレーブとして動作させ、SBI 割り込みを使用して I2C バスのリード/ライトを行うサンプルプログラムです。(評価ボード上の 2 本の I2C バス(SBI0 と SBI1)を接続します)
I2C スレーブアドレス: 0xB0
I2C スレーブ(SBI1)は I2C マスタ(SBI0)から“TOSHIBA”を受け取ります。

受信結果はデバッガ内の RAM 変数 gI2CrxData[]によって確認できます。

補足: このサンプルソフトを実行ために評価ボードを以下のように接続します。
PC1(SCL0)端子を PG1(SCL1)端子と接続します。
PC0(SDA0)端子を PG0(SDA1)端子と接続します。

6-12 SIO/UART

6-12-1 UART0/UART2 間の転送

本サンプルプログラムでは、データを UART0 から送信し、そのデータを UART2 で受信します。

UART0 からの送信データ: 文字列“TMPM384”
UART2 での受信データを変数 RxBuffer に格納します。関数 ResetIdx()の前にブレークポイントを設定すると、ブレークポイントにてプログラムが停止したときに変数 RxBuffer には文字列 "TMPM384"が保存されています。

UART 設定:
通信チャネル: UART0 と UART2
データ長 : 8 bits
パリティ : なし
ストップ : 1 bit
ボーレート : 115200bps
フロー制御 : なし

6-12-2 リターゲット

C 言語の標準入出力ライブラリの標準入力(stdin)、標準出力(stdout)をターゲットのハードウェアに合わせて変更することをリターゲットと呼びます。
この例は、標準入力と標準出力を、共にUARTに設定します。アプリケーションからprintf()関数を使ってシリアルポートから出力を行うサンプルプログラムです。

6-12-3 UART FIFO

UART0 から" TMPM3841"というデータを FIFO を使用して UART1 へ送信し、同時に UART1 から" TMPM3842"というデータを FIFO を使用して UART0 へ送信するサンプルプログラムです。
ResetIdx()関数の前にブレークポイントを設定しておくと、RxBuffer=” TMPM3842”と RxBuffer1=" TMPM3841"となった場合にブレークポイントで停止します。

6-12-4 SIO

この例では、TMPM384 の SIO モジュールを使用して同期式の送受信を行います。
SIO のチャネル 0 とチャネル 1 を使用し、これらのチャネル間で同期式データ送信を行います。
(TXD0 と RXD1、TXD1 と RXD0、sclk0 と sclk1 を接続してください)
*補足: この例では、スレーブ側を最初に動作させ、その後、マスタ側を動作させてください。

6-13 SSP

6-13-1 SSP0 から SSP1 への DMAC 転送

この例は、SSP0 を SPI マスタ、SSP1 を SPI スレーブに設定し、DMAC 転送を行うサンプルプログラムです。(評価ボード上の 2 本の SSP バス(SSP0 と SSP1)を接続します)

補足: 評価ボードを以下のように変更します。

	SSP0 端子	内容	SSP1 端子
1	PC3/SP0FSS , pin 18	PC3 と PN3 を接続します。 (pin 18 - 81)	PN3/SP1FSS , pin81
2	PC2/SP0CLK, pin 19	PC2 と PN2 を接続します。 (pin 19 - 67)	PN2/SP1CLK , pin67
3	PC1/SP0DI, pin 20	PC1 と PN0 を接続します。 (pin 20 - 65)	PN1/SP1DI, pin66
4	PC0/SP0DO, pin 21	PC0 と PN1 を接続します。 (pin 21 - 66)	PN0/SP1DO, pin65

6-13-2 SSP0 セルフループバック

この例は、送信側はデータを送信し、受信側は受信データをチェックするサンプルプログラムです。

6-14 TMRB

6-14-1 汎用タイマ

この例は、MCU のタイマを使って、汎用タイマを実現するサンプルプログラムです。
タイマ設定は 1ms 周期です。
このタイマを使い 1s (500ms で ON、500ms で OFF) 間隔で LED 点滅を行います。

6-14-2 PPG 波形出力

この例は、SW1 を使い、デューティ可変の PPG(プログラマブル矩形波)の波形を出力するサンプルプログラムです。
PPG 波形は、PA5(TB6OUT)をオシロスコープに接続することで確認できます。
デューティは 5 段階(10%, 25%, 50%, 75%, 90%)に変更できます。
電源投入すると PPG モードに入り、PPG 波形出力を開始します。
キーを押すことでデューティーを変更します。
10% → 25% → 50% → 75% → 90% → 10%

6-15 VLTD

この例は、VLTDによって電圧状態を検出するサンプルプログラムです。
電圧が検出電圧より低い場合、LED3を点灯し、UARTに“Low voltage!”を表示します。(デバッグモード時)
電圧が検出電圧より高い場合、LED3を消灯し、“High voltage!”を表示します。(デバッグモード時)

6-16 WDT

ウォッチドッグタイマは、高速クロックが停止する STOP モードでは使用できません。リセット後ウォッチドッグタイマは有効となり、SystemInit()関数で無効になります。

ドライバ使用方法ステップ:

1. 検出時間の設定とカウンターオーバーフロー時の動作としての WDT 割り込み設定を行い、WDT を初期化します。
2. 2 つの WDT 用サンプルがあり、DEMO2 はマクロ定義にて切り替えられます。

DEMO1:

タイマーオーバーフロー時に NMI 割り込みを発生し、WDT をクリアします。

DEMO2:

ウォッチドッグタイマ制御レジスタにクリアコードを書き込み、ウォッチドッグタイマカウンタをクリアします。

6-17 ENC

この例では、ENC ドライバを使用してホールマウスの回転を検知します。

動作シーケンス:

1. ホイールマウスエンコーダーの端子 A(図 1 参照)と端子 12 (ENCA0)を接続します。
2. ホイールマウスエンコーダーの端子 B(図 1 参照)と端子 11 (ENCB0)を接続します。
3. USB マウスと PC を接続し、ホイールマウスエンコーダーの Com(図 1)を 5V と接続します。
4. プログラムを実行します(LED1 が点滅します)。
5. マウスのホイールを前に回転すると LED4 はより短い間隔で点滅します。点滅間隔が最も短くなると、点滅間隔は初めの間隔に戻ります。
6. マウスのホイールを後ろに回転すると LED4 はより長い間隔で点滅します。点滅間隔が最も長くなると、点滅間隔は初めの間隔に戻ります。

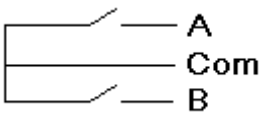


図 1

6-18 PMD

この例は、PMD ドライバを使用して位相出力を行います。

動作シーケンス:

1. プロジェクトを開き、DEMO_U_PHASE、または DEMO_3_PHASE と定義されたサンプルを選択します。プログラムを実行します。
2. EMG1 (pin 45)を DVSS (pin 116)とプルダウン接続します。(LED4 が点灯します)これは EMG 保護状態であることを意味します。
3. EMG1 (pin 45)を DVDD5 (pin 17)とプルアップ接続します。(LED4 が消灯します)これは PMD が通常動作であることを意味します。
4. UO1 (pin 29) をオシロスコープに接続し、波形を確認します。
5. XO1 (pin 30) をオシロスコープに接続し、波形を確認します。
6. VO1 (pin 31) をオシロスコープに接続し、波形を確認します。
7. YO1 (pin 32) をオシロスコープに接続し、波形を確認します。
8. WO1 (pin 43) をオシロスコープに接続し、波形を確認します。
9. ZO1 (pin 44) をオシロスコープに接続し、波形を確認します。

DEMO_U_PHASE 定義を選択した場合、位相波形は同じです。
UO1 のデューティは 25%、出力電圧は 5V、周期は 820us です。XO1 は UO1 を反転させた形です。VO1 と WO1 は UO1 と同じで、YO1 と ZO1 は XO1 と同じです。

DEMO_3_PHASE 定義を選択した場合、位相波形は異なります。
UO1 のデューティは 25%、出力電圧は 5V、周期は 820us です。XO1 は UO1 を反転させた形です。
VO1 のデューティは 50%、出力電圧は 5V、周期は 820us です。YO1 は VO1 を反転させた形です。
WO1 のデューティは 75%、出力電圧は 5V、周期は 820us です。ZO1 は WO1 を反転させた形です。

7 ソフトウェア

本ソフトウェアは、TMPM384 MCU の主要機能を TMPM384 評価ボード上で動作確認するためのサンプルプログラムです。

サンプルプログラムの実装には、最新のドライバーソフト、および IAR EWARM、または KEIL MDK をご使用ください。機能ごとにプロジェクトを作成してください。

ワークスペース構造とプロジェクト名は、以下の通りです:

IAR EWARM:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm384_wdt_int.c
        ├─TX03_CMSIS
        │   │   system_TMPM384.c
        │   │   system_TMPM384.h
        │   │   TMPM384.h
        │   └─startup
        │       startup_TMPM384.s
        │
        ├─TX03_Periph_Driver
        │   ├─inc
        │   │   tmpm384_wdt.h
        │   │   tmpm384_gpio.h
        │   │   tx03_common.h
        │   │
        │   └─src
        │       tmpm384_wdt.c
        │       tmpm384_gpio.c
        └─TMPM384-EVAL
                led.c
                led.h
```

KEIL MDK:

```
├─WDT_NMI
        ├─APP
        │   main.c
        │   tmpm384_wdt_int.c
        │
        ├─TX03_CMSIS
        │   │   system_TMPM384.c
        │   │   system_TMPM384.h
        │   │   TMPM384.h
        │   └─startup
        │       startup_TMPM384.s
        │
        ├─TX03_Periph_Driver
        │    tmpm384_wdt.c
        │    tmpm384_gpio.c
        │
        └─TMPM384-EVAL
              led.c
```

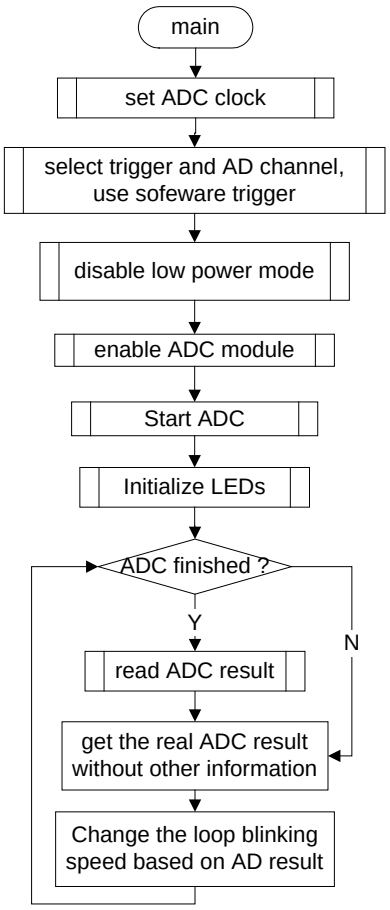
7-1 ADC

7-1-1 例: ADC データリード

TX03 ペリフェラルドライバ(ADC, GPIO)を用いたサンプルプログラムです。

この例では以下を行います。
1. ADC 設定と初期化を行います。
2. AD 変換の開始と AD 変換結果の読み出しを行います。
3. ループ点滅スピードの調整に AD 変換結果を使用ます。

- **フローチャート:**



- **サンプルプログラムのコードと説明**

最初に AD コンバータクロックの選択、チャネルの選択、ソフトウェア起動モードの許可、消費電力削減の許可を行います。

```
/* 1. set ADC clock */
```

```
        ADC_SetClk(ADC_HOLD_FIX, ADC_FC_DIVIDE_LEVEL_NONE);

    /* 2. select trigger and AD channel, this time we use sofeware trigger, */
    /*       remember to input with macro TRG_ENABLE() */
    ADC_SetSWTrg(ADC_REG0, TRG_ENABLE(ADC_CHANNEL));

    /* 3. to let ADC module work, we must disable low power mode */
    ADC_SetLowPowerMode(DISABLE);
```

ADC 動作を許可し、AD 変換を開始します。

```
    /* 4. enable ADC module */
    ADC_Enable();

    /* 5. now start ADC */
    ADC_Start(ADC_TRG_SW);
```

AD 変換を開始すると、AD 変換終了フラグがセットされるまで待ちます。AD 変換フラグがセットされると、AD 変換結果を取得し、LED 点滅間隔を変更します。

```
    /* check ADC module state */
    adcState = ADC_GetConvertState(ADC_TRG_SW);

    if (adcState == DONE) {
        /* read ADC result when it is finished */
        adResult = ADC_GetConvertResult(ADC_REG0);

        /* Get the real ADC result without other information */
        /* "/256" is to limit the range of AD value */
        timeUp = 16U - adResult.Bit.ADResult / 256U;
    /* use 'timeUp' above to adjust the loop blinking speed */

    /* software trigger, need to trig it again */
        ADC_Start(ADC_TRG_SW);
    } else {
        /* Do nothing */
    }
```

7-2 CG

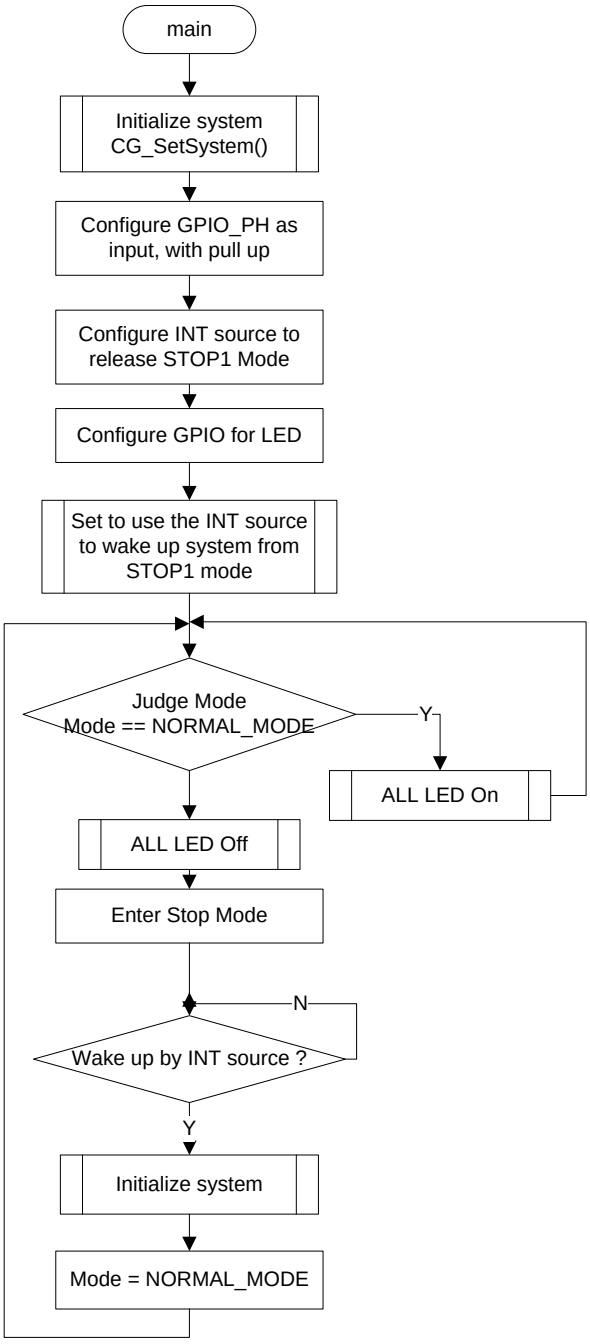
7-2-1 例: パワーモード変更

TX03 ペリフェラル・ドライバ(CG, GPIO,UART)を用いたサンプルプログラムです。

以下の例が含まれます:

1. 基本的な CG 動作の設定
2. NORMAL モードと SLEEP モードの切り替え方法

- **フローチャート**



- **サンプルプログラムのコードと説明**

以下は NORMAL モードと SLEEP モードの切り替えを行うサンプルプログラムです。

```
void CG_SetSystem(void)
{
    /* Set fgear = fc/2 */
    CG_SetFgearLevel(CG_DIVIDE_2);
    /* Set fperiph to fgear */
    CG_SetPhiT0Src(CG_PHIT0_SRC_FGEAR);
```

```
    CG_SetPhiT0Level(CG_DIVIDE_64);
    /* select external oscillator */
    CG_SetFoscSrc(CG_FOSC_OSC1);
    CG_SetPortM(CG_PORTM_AS_HOSC);
    /* Enable high-speed oscillator */
    CG_SetFosc(CG_FOSC_OSC1,ENABLE);
    /* Set low power consumption mode stop */
    CG_SetSTBYMode(CG_STBY_MODE_STOP);
    /* Set pin status in stop mode to "active" */
    CG_SetPinStateInStopMode(ENABLE);
    /* Set up pll and wait for pll to warm up, set fc source to fpll */
    CG_EnableClkMulCircuit();
}
```

スイッチ、LED、外部入力端子の設定

GPIO をスイッチ、LED、外部入力割り込み機能に設定します。

```
        /* Initialize system */
        CG_SetSystem();

        /* Configure GPIO */
        GPIO_SetInput(GPIO_PH, GPIO_BIT_0);      /* Set port H pin0 to input */
        GPIO_EnableFuncReg(GPIO_PH, GPIO_FUNC_REG_1, GPIO_BIT_0);
        /* Set port H pin0 for INT0 */
        GPIO_SetInput(GPIO_PH, GPIO_BIT_1);      /* Set port H pin1 to input */
        /* Configure GPIO to LED */
        LED_Init();
```

スタンバイ解除のための外部割り込みの設定

スタンバイ解除のために INT0 の設定を行います。割り込み保留要求レジスタをクリアし、INT0 を有効にします。

```
        /* Disable interrupts */
    __disable_irq();
    CG_ClearINTReq(CG_INT_SRC_0);
    /* Set external interrupt to wake up system */
    CG_SetSTBYReleaseINTSrc(CG_INT_SRC_0,
                                    CG_INT_ACTIVE_STATE_FALLING, ENABLE);
    NVIC_ClearPendingIRQ(INT0_IRQn);
    NVIC_EnableIRQ(INT0_IRQn);
    __enable_irq();
```

STOP モード設定

STOP モードに入る設定を行います。ウォームアップ時間を設定し、__WFI()命令を使用して STOP モードに入ります。

```
/* CG_NormalToStop */
void CG_NormalToStop(void)
{
    /* Set CG module: Normal ->Stop mode */
    CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC1,CG_WUODR_EXT);
    /* Enter stop mode */
    __WFI();
}
```

マルチクロック回路の許可

まず PLL を設定します。そしてウォームアップ時間を設定し、ウォームアップ時間が完了するまで待ちます。その後、fPLL を fc ソースとして設定します。

```
    Result CG_EnableClkMulCircuit(void)
```

```
{
	Result retval = ERROR;
	WorkState st = BUSY;
	retval = CG_SetPLL(ENABLE);
	if (retval == SUCCESS) {
		/* Set warm up time */
		CG_SetWarmUpTime(CG_WARM_UP_SRC_OSC1,CG_WUODR_PLL);
		CG_StartWarmUp();

		do {
			st = CG_GetWarmUpState();
		} while (st != DONE);

		retval = CG_SetFcSrc(CG_FC_SRC_FPLL);
	} else {
		/*Do nothing */
	}

	return retval;
}
```

7-3 DMAC

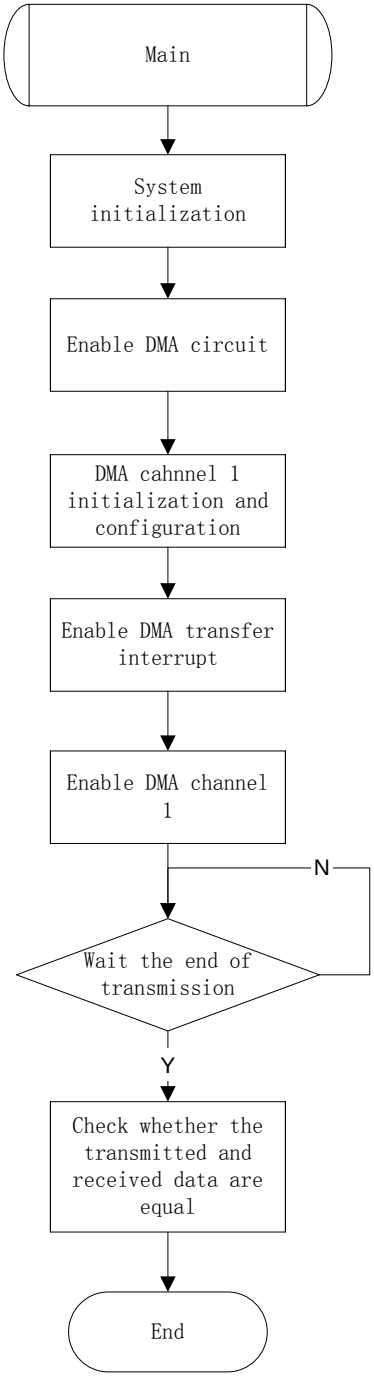
7-3-1 例: メモリから周辺回路

TX03 ペリフェラルドライバ(DMAC)を使用したサンプルプログラムです。

以下の例が含まれます:

1. DMAC の初期化を行います。
2. DMAC 転送を行います。

- **フローチャート**



- **サンプルプログラムのコードと説明**

データ転送のために DMAC ユニットのチャネル 1 の初期化と設定を行います。

```
DMAC_InitTypeDef DMAC_InitStruct;

DMAC_InitStruct.TxDirection = DMAC_MEMORY_TO_MEMORY;
DMAC_InitStruct.SrcAddr = (uint32_t) SRC_Buffer;
```

```
DMAC_InitStruct.DstAddr = (uint32_t) DST_Buffer;
DMAC_InitStruct.SrcIncrementState = ENABLE;
DMAC_InitStruct.DstIncrementState = ENABLE;
DMAC_InitStruct.SrcBitWidth = DMAC_WORD;
DMAC_InitStruct.DstBitWidth = DMAC_WORD;
DMAC_InitStruct.SrcBurstSize = DMAC_4_BEATS;
DMAC_InitStruct.DstBurstSize = DMAC_4_BEATS;
DMAC_InitStruct.TxSize = BUFFER_SIZE;
DMAC_InitStruct.TxINT = ENABLE;
```

DMAC ユニットの許可、DMA チャネルの初期化を行います。その後、DMAC ユニットのチャネル 1 から転送を開始します。

```
DMAC_Enable();
DMAC_Init(myChannel, &DMAC_InitStruct);
DMAC_TxINTConfig(myChannel, DMAC_INT_TX_END, ENABLE);
DMAC_SetDMAChannel(myChannel, ENABLE);
```

DMAC 転送が終わるまで待ちます。

```
while (TxEndFlag != DONE) {
        /* Do nothing */
}
```

DMAC 転送終了時に ISR の DMAC ユニットにフラグがセットされます。

```
state = DMAC_GetINTReq();

if (state.Bit.CH1_INTReq == 1U) {
    tmp = DMAC_GetTxINTReq(DMAC_CHANNEL_1);
    if (tmp == DMAC_TX_END_REQ) {
        TxEndFlag = DMAC_GetChannelTxState(DMAC_CHANNEL_1);
        DMAC_ClearTxINTReq(DMAC_CHANNEL_1, DMAC_INT_TX_END);
    } else {
        /* Do nothing */
    }
} else {
    /* Do nothing */
}
```

7-4 FLASH

7-4-1 例: FLASH スワップ

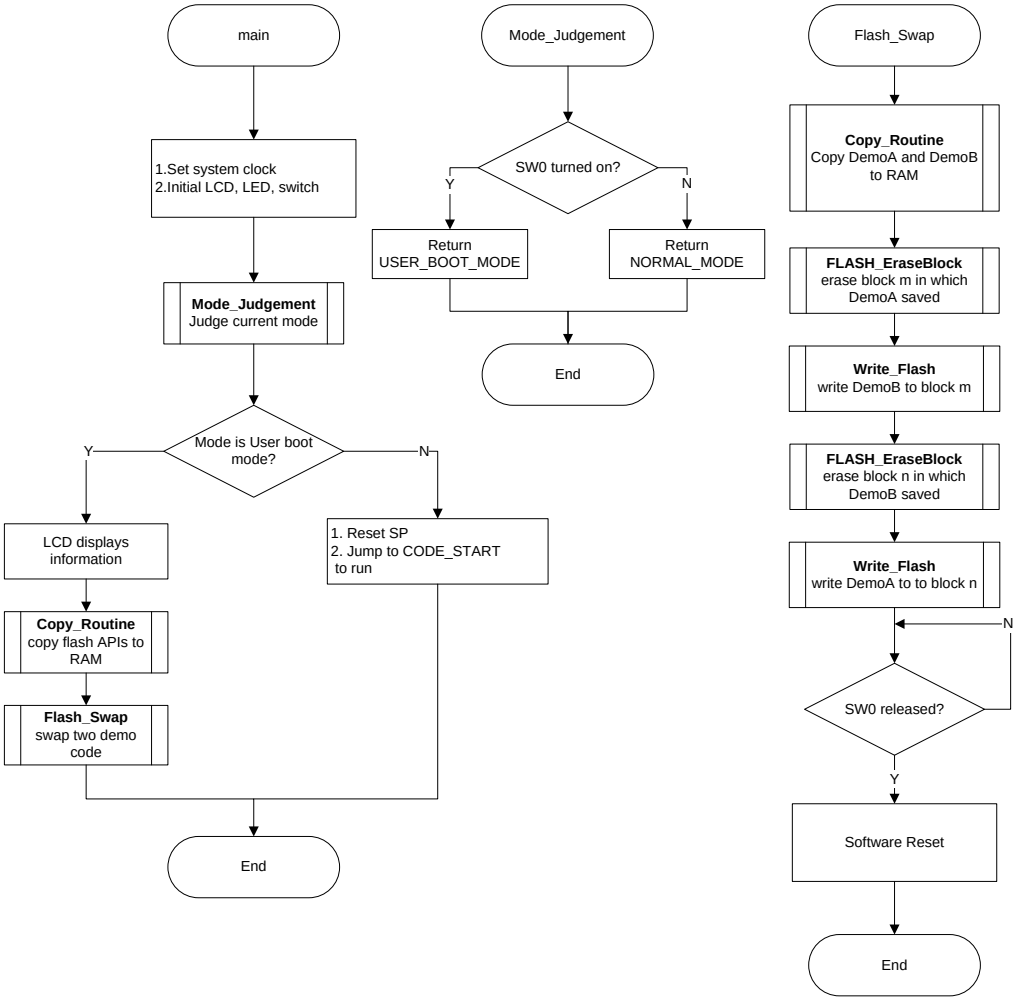
TX03 ペリフェラルドライバ(Flash, GPIO)を使用したサンプルプログラムです。

以下の例が含まれます:

1. FLASH メモリのオンボードプログラミング (書き込み/消去)

2. 動作モード:シングルチップモード (ノーマルモード、ユーザーブートモード)
3. ユーザーブートモードを使用し、Flash メモリのコードを更新。

- **フローチャート**



- **サンプルプログラムのコードと説明**

まず LED, LCD, SW0 を初期化します。リセット時に現在のモード判定を SW0 で行います。

```
    LED_Init();
    SW_Init();
    LCD_Configuration();
```

リセット後にどのモードになっているかの判断を行うために、Mode_Judgement()関数をコールします。

```
uint8_t Mode_Judgement(void)
{
    return (SW_Get(SW0) == 1U) ? USER_BOOT_MODE : NORMAL_MODE;
}
```

SW0 がリセット時に ON でない場合、動作モードはノーマルモードになります。プログラムは SP を初期化し、アドレス“CODE_START”へジャンプします。

```
#if defined ( __CC_ARM      )          /* RealView Compiler */
```

```
        ResetSP();                      /* reset SP */
#elif defined ( __ICCARM__ )        /* IAR Compiler */
        asm("MOV R0, #0");          /* reset SP */
        asm("LDR SP, [r0]");
#endif
        SCB->VTOR = DEMO_START_ADDR;        /* redirect vector table */
        startup = CODE_START;
        startup();                      /* jump to code start address to run */
```

リセット時に SW0 が ON の場合、動作モードはユーザブートモードに移行し、LCD に情報を表示します。フラッシュメモリ上で実行しながらフラッシュメモリ自身を消去/プログラムできないため、プログラムはフラッシュ操作ルーチンをアドレス“FLASH_API_ROM”からアドレス“FLASH_API_RAM”へ ROM->RAM 転送します。

```
        LCD_Display("User Boot Mode", "");          /* enter user boot mode */
        LCD_Display("RAM transferring", "......");

        Copy_Routine(FLASH_API_RAM, FLASH_API_ROM, SIZE_FLASH_API);
       /* copy flash API to RAM */

        LCD_Display("Flash swapping", "......");
```

Flash 動作用 API を RAM にコピー後、プログラムは Flash_Swap()関数にジャンプします。この関数は、上記の Copy_Routine() を使用し、Flash メモリーから RAM にコピーされます。

Flash_Swap() 関数では、まずサンプル A とサンプル B のプログラムを Flash ROM から RAM にコピーします。次に、Flash ROM の消去/書き込みを行うため、関数 FC_EraseBlock()と Write_Flash()をコールします。サンプル A とサンプル B のプログラムは Flash ROM 内にて差し替えます。

```
    Copy_Routine(DEMO_A_RAM,  DEMO_A_FLASH,  SIZE_DEMO_A);              /*
copy A to RAM */
    Copy_Routine(DEMO_B_RAM,  DEMO_B_FLASH,  SIZE_DEMO_B);              /*
copy B to RAM */

    if (FC_SUCCESS == FC_EraseBlock((uint32_t) DEMO_A_FLASH)) { /* erase A */
        /* Do nothing */
    } else {
        return ERROR;
    }

    if    (FC_SUCCESS    ==    Write_Flash(DEMO_A_FLASH,    DEMO_B_RAM,
SIZE_DEMO_B)) {       /* write B to A */
        /* Do nothing */
    } else {
        return ERROR;
    }

    if (FC_SUCCESS == FC_EraseBlock((uint32_t) DEMO_B_FLASH)) { /* erase B */
        /* Do nothing */
    } else {
        return ERROR;
    }

    if    (FC_SUCCESS    ==    Write_Flash(DEMO_B_FLASH,    DEMO_A_RAM,
SIZE_DEMO_A)) {       /* write A to B */
```

```
        /* Do nothing */
    } else {
        return ERROR;
    }
```

サンプル A とサンプル B のスワップ動作が終了したことを示す情報を LCD に表示します。SW0 が OFF の場合、SCB->AIRCR レジスタにて自動的にソフトウェアリセットを行います。そして、再び NORMAL モードで動作します。サンプル A とサンプル B が入れ替わったため、アドレス “CODE_START”はサンプル B の開始アドレスとなり、サンプル B が実行されます。(LED1 が点滅します)

```
    LCD_Display("Finished", "Restart.....");


    while (SW_Get(SW0) == 1U) {
    }

    reg_value = SCB->AIRCR;         /* software reset */
    reg_value = (uint32_t) 0x05FA0001;
    SCB->AIRCR = reg_value;
```

Flash メモリ動作関数 FC_EraseBlock() は自動的に指定されたブロックを消去します。このブロックは最初に引数 “BlockAddr” で指定します。まず、この関数で引数 “BlockAddr” を確認します。次に、Flash ドライバー FC_GetBlockProtectState() を使用し、指定されたブロックがプロテクトされているか確認します。

```
    if (ENABLE == FC_GetBlockProtectState(BlockNum)) {
        retval = FC_ERROR_PROTECTED;
    }
```

ブロックにプロテクトがかかっている場合、“FC_ERROR_PROTECTED” を返します。それ以外の場合は、自動的ブロック削除命令を送り、ブロックを削除します。

```
        *addr1 = (uint32_t) 0x000000AA; /* bus cycle 1 */
        *addr2 = (uint32_t) 0x00000055; /* bus cycle 2 */
        *addr1 = (uint32_t) 0x00000080; /* bus cycle 3 */
        *addr1 = (uint32_t) 0x000000AA; /* bus cycle 4 */
        *addr2 = (uint32_t) 0x00000055; /* bus cycle 5 */
        *BA = (uint32_t) 0x00000030;       /* bus cycle 6 */
```

次に、削除が終了すると、Flash ドライバ FC_GetBusyState() を用いモニターリングを行います。同時に、タイムアウトカウンタを使用し、動作が指定時間を越えていないか確認します。

```
        while (BUSY == FC_GetBusyState()) {       /* check if FLASH is busy with
overtime counter */
            if (!(counter--)) { /* check overtime */
                retval = FC_ERROR_OVER_TIME;
                break;
            } else {
                /* Do nothing */
            }
        }
```

関数 Write_Flash() は FC_WritePage () を呼び出し、自動的に 1 ページにデータを書き込みます。この動作は、自動ページプログラム命令を除き、基本的に FC_EraseBlock () と同じです。

```
*addr1 = (uint32_t) 0x000000AA; /* bus cycle 1 */
*addr2 = (uint32_t) 0x00000055; /* bus cycle 2 */
*addr1 = (uint32_t) 0x000000A0; /* bus cycle 3 */
for (i = 0U; i < FC_PAGE_SIZE; i++) {      /* bus cycle 4~67 */
      *addr3 = *source;
      source++;
}
```

7-5 GPIO

TX03 ペリフェラルドライバ GPIO)を用いたサンプルプログラムです。

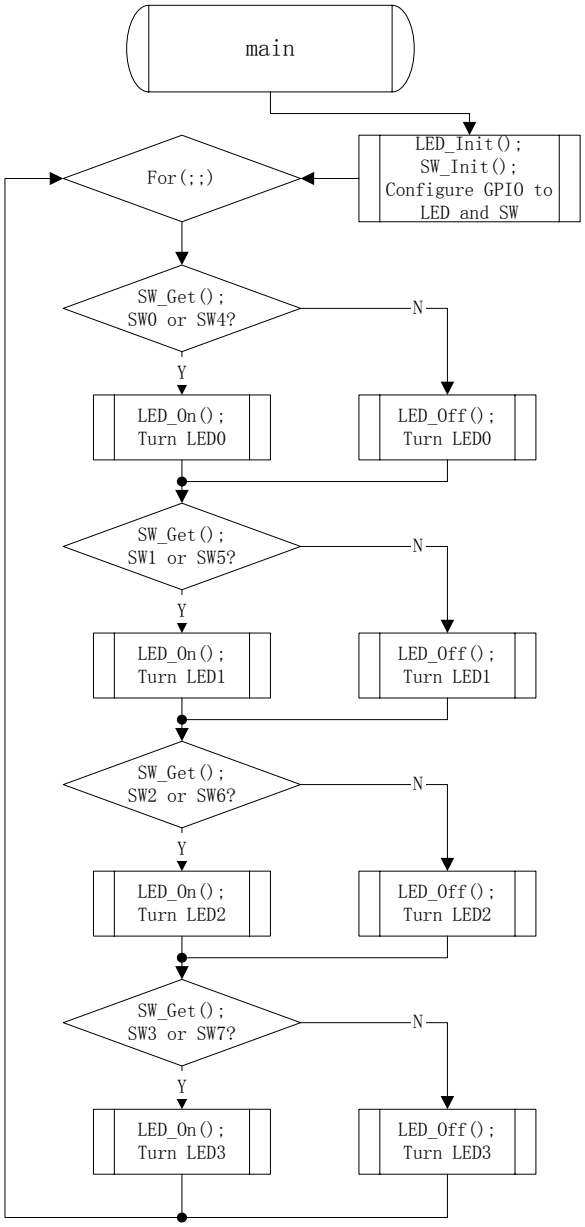
以下の例が含まれます:

1. GPIO の初期化
2. GPIO へのデータ書き込み
3. GPIO からのデータ読み出し

SW, LED, LCD の端子接続:

端子	機能
PH0	SW0, SW4
PH1	SW1, SW5
PH2	SW2, SW6
PH3	SW3, SW7
PT0	LED0
PT1	LED1
PT2	LED2
PT3	LED3
PH4	SWS

- **フローチャート:**



- **サンプルプログラムのコードと説明:**

まず GPIO_SetOutput(GPIO_Port GPIO_x, uint8_t Bit_x)を使用して GPIO を LED に設定します。そして GPIO_SetInput(GPIO_Port GPIO_x, uint8_t Bit_x)を使用して GPIO をキーに設定します。以下に例を示します。

```
/* LED0~LED3 */
GPIO_SetOutput(LED_DATA_PORT, 0x0FU);
GPIO_WriteData(LED_DATA_PORT, 0x00U);

/* SW0~7, SWS*/
GPIO_SetInput(SW_PORT, W04_BIT|SW15_BIT|SW26_BIT|SW37_BIT);
GPIO_SetOutput(SWS_PORT, SWS_BIT);
```

for(; ;)ループの中で、LED サンプルを実行します: スイッチ状態をリードし、LED オンや LED オフを制御します。

GPIO_WriteDataBit(GPIO_Port GPIO_x, uint8_t Bit_x, uint8_t BitValue)を使用して LED を点灯します。

```
uint32_t tmp;
tmp = GPIO_ReadData(LED_DATA_PORT);
tmp |= led;
GPIO_WriteData(LED_DATA_PORT, tmp);
```

GPIO_WriteDataBit(GPIO_Port GPIO_x, uint8_t Bit_x, uint8_t BitValue)を使用して LED を消灯します。

```
uint32_t tmp;
tmp = GPIO_ReadData(LED_DATA_PORT);
tmp &= ~led;
GPIO_WriteData(LED_DATA_PORT, tmp);
```

7-6 OFD

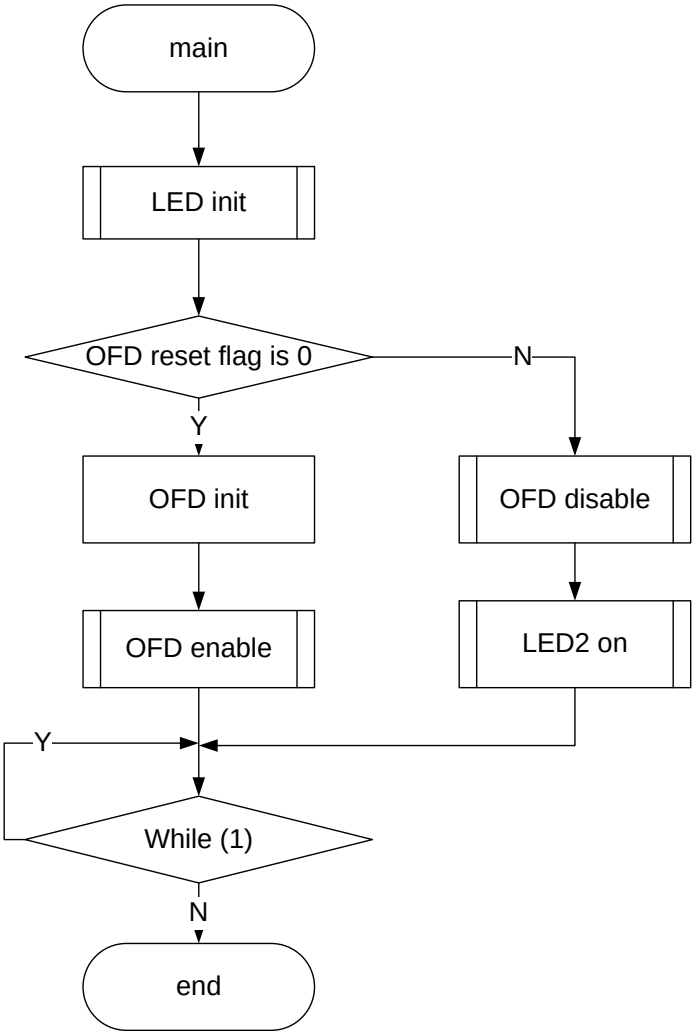
7-6-1 例: スタートアップ

TX03 ペリフェラルドライバ(OFD, CG, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます:

1. OFD の初期化(検知周波数範囲を OFD に設定します)
2. OFD リセットフラグの確認
3. OFD の有効／無効

- **フローチャート**



- **サンプルプログラムのコードと説明**

まず、LED を初期化します。

```
    Init_Led();
```

OFD のリセットフラグを確認します。

```
    resetFlag = CG_GetResetFlag();
    if (resetFlag.Bit.OFDReset == 0)
```

このフラグが設定されていない場合、外部発振器の許可と検知周波数を設定して OFD を初期化します。

以下は外部発振器を許可する処理です。

```
    void ChangeToEHOSC(void)
    {
        /* Use the external oscillation */
        TSB_CG->OSCCR &= 0x000FFFFFUL;
        TSB_CG->OSCCR |= 0xFFF00000UL;          /*Set the warm up time WUODR[13:2]*/
```

```
        TSB_CG_OSCCR_HOSCON = 1U;     /*Set gpio port as X1/X2 for external
oscillator */
        TSB_CG_OSCCR_XEN1 = 1U;         /*Enable external oscillator */
        TSB_CG_OSCCR_WUPSEL2 = 1U;
        TSB_CG_OSCCR_WUPSEL1 = 0U;    /*Select warm-up clock */
        TSB_CG_OSCCR_WUEON = 1U;       /*Start warm up */
        while (TSB_CG_OSCCR_WUEF) {}/* Warm-up */
        TSB_CG_OSCCR_OSCSEL = 1U;      /*Use the external oscillation */
        TSB_CG_OSCCR_XEN2 = 1U;         /*Enable internal oscillator for ofd */
    }
```

OFD の設定を行うため、まずレジスタ書き込み許可コードを設定します。

```
    OFD_SetDetectionFrequency(OFD_HIGHER_COUNT,
                                                    OFD_LOWER_COUNT);
```

DEMO2 を定義すると異常な周波数範囲が設定されます。

```
    OFD_SetDetectionFrequency(OFD_HIGHER_COUNT_ABNORMAL,
                                                OFD_LOWER_COUNT_ABNORMAL);
```

初期化し、その後、OFD を有効にします。

```
    OFD_Enable();
```

上記の設定を行い、その後 OFD は外部クロックを検知します。このクロックが検知周波数範囲を超える(OSC が乱れる、DEMO2 が定義されている)と OFD は周波数検知リセットを発生します。これにより OFD のリセットフラグが設定されます。
このフラグを検知すると、デフォルト内部発振器で MCU が実行を始め、OFD は無効となり、OFD リセットフラグに'1'がセットされます。

```
    OFD_Disable();
    OFDReset = 1U;
```

LED1 と LED2 はシステムクロックの状態を表示します。外部発振器にて発振している場合は LED1 を点滅し、内部発振器にて発振している場合は、LED2 を点滅します。

```
    while (1U) {
        if (OFDReset == 1U) {
            LED_On(LED2);
            Delay();
            LED_Off(LED2);
            Delay();
        } else {
            LED_On(LED1);
            Delay();
            LED_Off(LED1);
            Delay();
        }
    }
```

7-7 IGBT

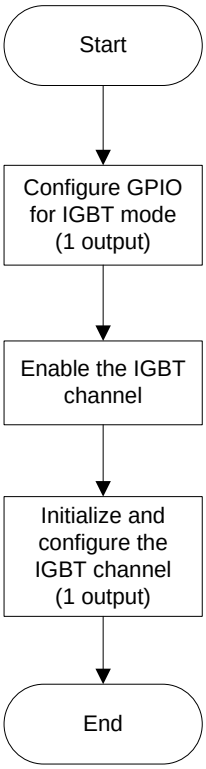
7-7-1 例: シングル PPG 波形出力

TX03 ペリフェラルドライバ(IGBT, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます:

1. IGBT タイマの設定と初期化 (シングル PPG 出力)
2. IGBT タイマの EMG 保護周波数
3. IGBT タイマ状態と EMG 割り込み状態の使用方法

- **フローチャート**



- **サンプルプログラムのコードと説明**

LED のコンフィギュレーションを行い、その後、IGBT_InitTypeDef 構造体の生成とデータ設定を行います。

```
IGBT_InitTypeDef myIGBT;
/* IGBT trigger start: falling edge start and active level is "Low" */
myIGBT.StartMode = IGBT_FALLING_TRG_START;
/* IGBT operation: continuous operation */
myIGBT.OperationMode = IGBT_CONTINUOUS_OUTPUT;
```

```
    /* IGBT stopping status: initial output satus and counter */
    myIGBT.CntStopState = IGBT_OUTPUT_INACTIVE;
    /* Trigger edge accept mode: Don't accept trigger during active level */
    myIGBT.ActiveAcceptTrg = DISABLE;
    /* Interrupt cycle: Every one cycle */
    myIGBT.INTPeriod = IGBT_INT_PERIOD_1;
    /* fperiph = fc = 10MHz*4 = 40MHz, T0 = fperiph = 40MHz, figbt = T0/2 = 20MHz */
    myIGBT.ClkDiv = IGBT_CLK_DIV_2;
    /* MTOUT00 initial state is Low, and active level is High */
    myIGBT.Output0Init = IGBT_OUTPUT_HIGH_ACTIVE;
    /* Disable MTOUT10 output */
    myIGBT.Output1Init = IGBT_OUTPUT_DISABLE;
    /* Trigger input noise elimination time: 240/fsys */
    myIGBT.TrgDenoiseDiv = IGBT_DENOISE_DIV_240;
    myIGBT.Output0ActiveTiming = 1U;
    myIGBT.Output0InactiveTiming = 1U + IGBT_PPG_PERIOD_50US / 2;
    /* Period: 50us */
    myIGBT.Period = IGBT_PPG_PERIOD_50US;
    /* The polarity of MTOUT0x at EMG protection: High-impedance */
    myIGBT.EMGFunction = IGBT_EMG_OUTPUT_HIZ;
    /* EMG input noise elimination time: 240/fsys */
    myIGBT.EMGDenoiseDiv = IGBT_DENOISE_DIV_240;
```

その後 IGBT チャネルと EMG 保護割り込みのの有効化を行い、初期設定前に EMG ステートを中断します。

```
    /* Enable IGBT and EMG interrupt */
    IGBT_Enable(IGBT0);
    NVIC_EnableIRQ(INTMTEMG0_IRQn);

    /* If the timer is still running, wait until it stops */
    do {
        counter_state = IGBT_GetCntState(IGBT0);
    } while (counter_state == BUSY);
    /* Cancel the EMG protection state */
    do {
        cancel_result = IGBT_CancelEMGState(IGBT0);
    } while (cancel_result == ERROR);
```

IGBT_CancelEMGState()のリターンパラメータが **SUCCESS** の場合、IGBT_Init()をコールし、IGBT の初期設定を完了します。

```
  IGBT_Init(IGBT0, &myIGBT);
```

開始トリガを受ける前に最初にソフトウェア開始コマンドを発行してください。

```
IGBT_SetSWRunState(IGBT0, IGBT_RUN);
```

その後、開始トリガが検出されると対応するポートより PPG 波形が出力されます。

もし GEMG の EMG 保護レベルがアクティブの場合、EMG 割り込みが発生します。LED1 を点灯し、タイマを停止します。

```
void INTMTEMG0_IRQHandler(void)
{
    /* If EMG protection, turn on the LED and stop the timer */
    LED_On(LED1);
    IGBT_SetSWRunState(IGBT0, IGBT_STOP);
}
```

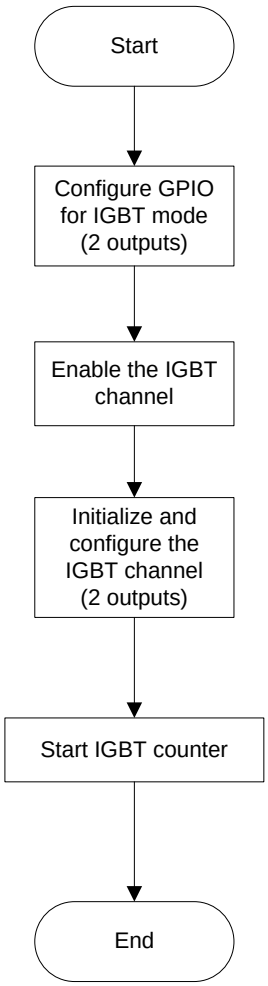
7-7-2 例: ダブル PPG 波形出力

TX03 ペリフェラルドライバ(IGBT, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます:

1. IGBT タイマの設定と初期化 (ダブル PPG 出力)
2. IGBT タイマの EMG 保護周波数
3. IGBT タイマ状態と EMG 割り込み状態の使用方法

- **フローチャート**



- **サンプルプログラムのコードと説明**

LED のコンフィギュレーションを行い、その後、IGBT_InitTypeDef 構造体の生成とデータ設定を行います。

```
IGBT_InitTypeDef myIGBT;
/* IGBT trigger start: command start */
myIGBT.StartMode = IGBT_CMD_START;
/* IGBT operation: continuous operation */
myIGBT.OperationMode = IGBT_CONTINUOUS_OUTPUT;
/* IGBT stopping status: initial output satus and counter */
myIGBT.CntStopState = IGBT_OUTPUT_INACTIVE;
/* Trigger edge accept mode: Don't accept trigger during active level */
myIGBT.ActiveAcceptTrg = DISABLE;
/* Interrupt cycle: Every one cycle */
myIGBT.INTPeriod = IGBT_INT_PERIOD_1;
```

```
    /* fperiph = fc = 10MHz*4 = 40MHz, T0 = fperiph = 40MHz, figbt = T0/2 = 20MHz */
    myIGBT.ClkDiv = IGBT_CLK_DIV_2;
    /* MTOUT00 initial state is Low, and active level is High */
    myIGBT.Output0Init = IGBT_OUTPUT_HIGH_ACTIVE;
    /* MTOUT10 initial state is Low, and active level is High */
    myIGBT.Output1Init = IGBT_OUTPUT_HIGH_ACTIVE;
    /* Trigger input noise elimination time: no use */
    myIGBT.TrgDenoiseDiv = IGBT_NO_DENOISE;
    myIGBT.Output0ActiveTiming = 1U;
    myIGBT.Output0InactiveTiming = 1U + IGBT_ACTIVE_PERIOD_20US;
    myIGBT.Output1ActiveTiming = myIGBT.Output0InactiveTiming +
                                 IGBT_DEAD_TIME_5US;
    myIGBT.Output1InactiveTiming = myIGBT.Output1ActiveTiming +
                                   IGBT_ACTIVE_PERIOD_20US;
    /* Period: 50us */
    myIGBT.Period = IGBT_PPG_PERIOD_50US;
    /* The polarity of MTOUT0x at EMG protection: High-impedance */
    myIGBT.EMGFunction = IGBT_EMG_OUTPUT_HIZ;
    /* EMG input noise elimination time: 240/fsys */
    myIGBT.EMGDenoiseDiv = IGBT_DENOISE_DIV_240;
```

その後 IGBT チャネルと EMG 保護割り込みのの有効化を行い、初期設定前に EMG ステートを中断します。

```
    /* Enable IGBT and EMG interrupt */
    IGBT_Enable(IGBT0);
    NVIC_EnableIRQ(INTMTEMG0_IRQn);

    /* If the timer is still running, wait until it stops */
    do {
        counter_state = IGBT_GetCntState(IGBT0);
    } while (counter_state == BUSY);
    /* Cancel the EMG protection state */
    do {
        cancel_result = IGBT_CancelEMGState(IGBT0);
    } while (cancel_result == ERROR);
```

IGBT_CancelEMGState()のリターンパラメータが **SUCCESS** の場合、IGBT_Init()をコールし、IGBT の初期設定を完了します。

```
  IGBT_Init(IGBT0, &myIGBT);
```

スイッチが ON の場合、開始コマンドを送信します。

```
  /* If switch is ON, start to run IGBT timer */
  if (SWITCH_ON) {
```

```
    IGBT_SetSWRunState(IGBT0, IGBT_RUN);
} else {
  /* Do nothing */
}
```

もしGEMGのEMG保護レベルがアクティブの場合、EMG割り込みが発生します。LED1を点灯し、タイマを停止します。

```
void INTMTEMG0_IRQHandler(void)
{
    /* If EMG protection, turn on the LED and stop the timer */
    LED_On(LED1);
    IGBT_SetSWRunState(IGBT0, IGBT_STOP);
}
```

7-8 RMC

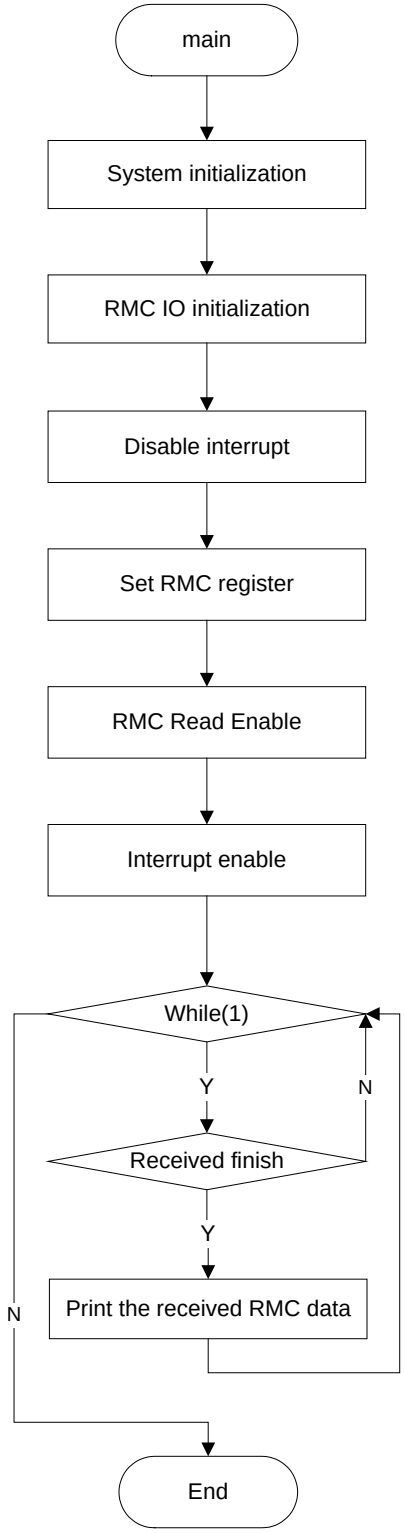
7-8-1 例: RMC 受信

TX03 ペリフェラルドライバ(RMC, UART, GPIO)を使用した簡単なサンプルソフトです。

以下の例が含まれます。

1. RMC 用 IO の初期化
2. RMC データの受信

- **フローチャート**


main
System initialization
RMC IO initialization
Disable interrupt
Set RMC register
RMC Read Enable
Interrupt enable
While(1)
Y
N
Received finish
Y
N
Print the received RMC data
End

- **サンプルプログラムのコードと説明**

まず main()にて myRMC structure 構造体を作成し、データフィールドにデータを入力します。

```
RMC_InitTypeDef myRMC;

myRMC.LeaderPara.MaxCycle = RMC_MAX_CYCLE;
myRMC.LeaderPara.MinCycle = RMC_MIN_CYCLE;
myRMC.LeaderPara.MaxLowWidth = RMC_MAX_LOW_WIDTH;
myRMC.LeaderPara.MinLowWidth = RMC_MIN_LOW_WIDTH;
myRMC.LeaderPara.LeaderDetectionState = ENABLE;
myRMC.LeaderPara.LeaderINTState = DISABLE;
myRMC.FallingEdgeINTState = DISABLE;
myRMC.SignalRxMethod = RMC_RX_IN_CYCLE_METHOD;
myRMC.LowWidth = RMC_TRG_LOW_WIDTH;
myRMC.MaxDataBitCycle = RMC_TRG_MAX_DATA_BIT_CYCLE;
myRMC.LargerThreshold = RMC_LARGER_THRESHOLD;
myRMC.SmallerThreshold = RMC_SMALLER_THRESHOLD;
myRMC.InputSignalReversedState = DISABLE;
myRMC.NoiseCancellationTime = RMC_NOISE_CANCELLATION_TIME;
```

次に、RMC チャネル 0 を初期化し、有効にします。

```
RMC_Enable(TSB_RMC0);
```

構造体には RMC の基本設定が含まれます。

```
RMC_Init(TSB_RMC0, &myRMC);
```

RMC チャネル 0 の受信を有効にする。

```
RMC_SetRxCtrl(TSB_RMC0, ENABLE);
```

割り込み INTRMCRX_IRQHandler()にて RMC チャネル 0 の割り込み要因を取得します。

```
RMC_INTFactor myRMC_INTFactor;
myRMC_INTFactor = RMC_GetINTFactor(TSB_RMC0);
```

RMC チャネル 0 のリーダー検出結果を取得します。

```
RMC_LeaderDetection myRMC_LeaderDetection;
myRMC_LeaderDetection = RMC_GetLeader(TSB_RMC0);
```

RMC チャネル 0 の受信データを取得します。次に、RMC チャネル 0 を初期化し、有効にします。

```
RMC_RxDataTypeDef myRMC_RxDataDef;
myRMC_RxDataDef = RMC_GetRxData(TSB_RMC0);
```

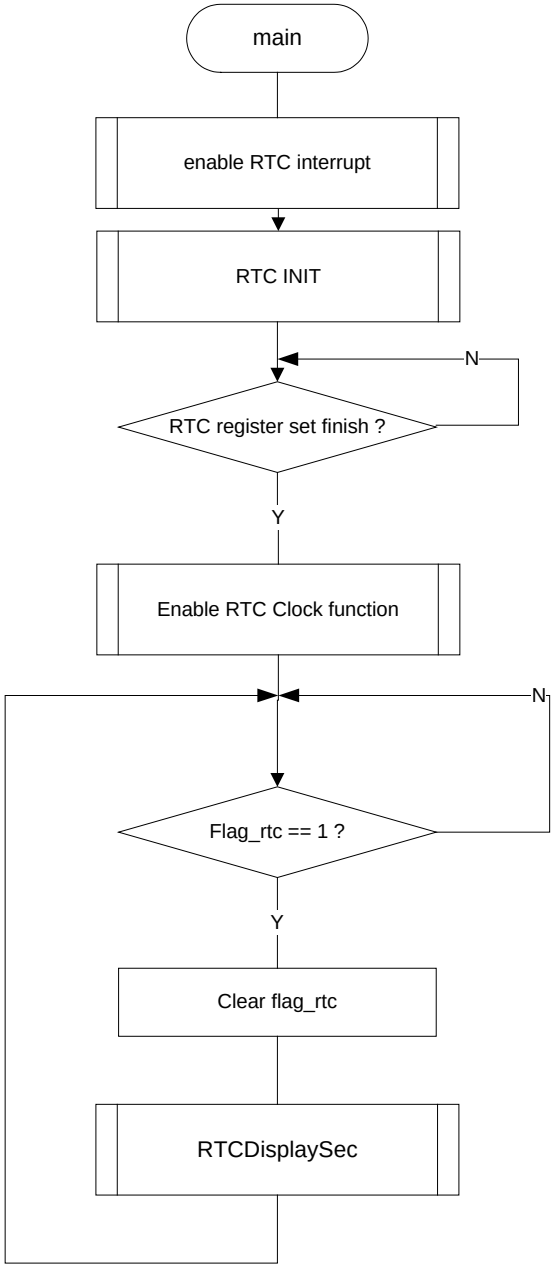
7-9 RTC

TX03 ペリフェラルドライバ(RTC, CG, GPIO)を使用した簡単なサンプルソフトです。

以下の例が含まれます。

1. RTC の初期化
2. RTC 秒の取得

- **フローチャート:**



- **サンプルプログラムのコードと説明:**

まず、RTC 割り込みによる SLEEP モードの解除設定を行います。

```
    CG_SetSTBYReleaseINTSrc(CG_INT_SRC_RTC,
                                    CG_INT_ACTIVE_STATE_FALLING, ENABLE);
```

RTC の初期化で、RTC_DateTypeDef と RTC_TimeTypeDef 構造を作成し、全データ項目を設定します。ここでは 2010/10/22 12:50:55, 24 時間表示フォーマットを初期設定としています。

```
    Date_Struct.LeapYear = RTC_LEAP_YEAR_2;
```

```
Date_Struct.Year = (uint8_t) 10U;
Date_Struct.Month = (uint8_t) 10U;
Date_Struct.Date = (uint8_t) 22U;
Date_Struct.Day = RTC_FRI;

Time_Struct.HourMode = RTC_24_HOUR_MODE;
Time_Struct.Hour = (uint8_t) 12U;
Time_Struct.Min = (uint8_t) 50U;
Time_Struct.Sec = (uint8_t) 55U;
```

クロックとアラーム機能を禁止します。

```
RTC_DisableClock();
RTC_DisableAlarm();
```

RTC 秒カウンタのリセット、1Hz 割り込みの許可、RTCINT の許可を行います。

```
RTC_ResetClockSec();
RTC_SetAlarmOutput(RTC_PULSE_1_HZ);
RTC_SetRTCINT(ENABLE);
```

RTC の時間と日付の値を設定します。

```
RTC_SetTimeValue(&Time_Struct);
RTC_SetDateValue(&Date_Struct);
```

上記項目を設定後、RTC レジスタ設定の終了を待ち、RTC 時計機能を許可します。その後、RTC クロック機能を許可します。

```
NVIC_EnableIRQ(INTRTC_IRQn);

/* waiting for RTC register set finish */
while (fRTC_1HZ_INT != 1U) {
      /* Do nothing */
}
fRTC_1HZ_INT = 0U;

/* Enable RTC Clock function */
RTC_EnableClock();
```

RTC 割り込みが 1 秒ごとに発生するよう設定を行い、その後、RTC 割り込み要求のクリアを行います。

```
fRTC_1HZ_INT = 1U;
CG_ClearINTReq(CG_INT_SRC_RTC);
```

RTC 割り込み発生後、第 2 番目の値がバイナリで LED ディスプレイに送信されます。
下記に、第 2 番目の値の取得方法と表示方法を表します。

```
Sec = RTC_GetSec();
tmp = Sec % 10U;
SegArrayDisplay(ledchar[tmp])
```

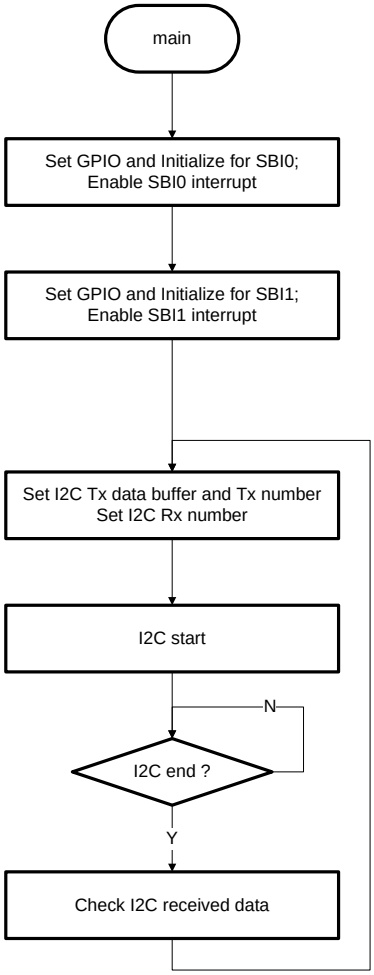
7-10 SBI

TX03 ペリフェラルドライバ(SBI, GPIO)を使用した簡単なサンプルソフトです。

以下の例が含まれます。

1. SBI 設定、I2C 初期化
2. I2C マスターによるデータプロセスの送信
3. I2C スレーブによるデータプロセスの受信

- **フローチャート**



- **サンプルプログラムのコードと説明**

まず、SBI0 と SBI1 の GPIO を I2C モードに設定します。

```
SBI0_IO_Configuration();
SBI1_IO_Configuration();
```

次に SBI0 チャネルの初期化と INTI2C をイネーブルにします。

```
myI2C.I2CSelfAddr = SELF_ADDR;
myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
myI2C.I2CACKState = ENABLE;
myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
SBI_Enable(TSB_SBI0);
SBI_SWReset(TSB_SBI0);
SBI_InitI2C(TSB_SBI0, &myI2C);
NVIC_EnableIRQ(INTSBI0_IRQn);
```

次に SB1 チャネルの初期化と INTSBI1 をイネーブルにします。

```
myI2C.I2CSelfAddr = SLAVE_ADDR;
```

```
    myI2C.I2CDataLen = SBI_I2C_DATA_LEN_8;
    myI2C.I2CACKState = ENABLE;
    myI2C.I2CClkDiv = SBI_I2C_CLK_DIV_328;
    SBI_Enable(TSB_SBI1);
    SBI_SWReset(TSB_SBI1);
    SBI_InitI2C(TSB_SBI1, &myI2C);
    NVIC_EnableIRQ(INTSBI1_IRQn);
```

上記設定を行った後、I2C 送信開始します。
I2C 受信バッファをクリアし、SBI TX バッファーとバッファ長を設定します。その後 RX バッファーをクリアします。

```
        /* Initialize TRx buffer and Tx length */
    case MODE_SBI_I2C_INITIAL:
        gI2CTxDataLen = 7U;
        gI2CTxData[0] = gI2CTxDataLen;
        gI2CTxData[1] = 'T';
        gI2CTxData[2] = 'O';
        gI2CTxData[3] = 'S';
        gI2CTxData[4] = 'H';
        gI2CTxData[5] = 'I';
        gI2CTxData[6] = 'B';
        gI2CTxData[7] = 'A';
        gI2CWCnt = 0U;
        for (gICnt = 0U; gICnt < 8U; gICnt++) {
            gI2CRxData[gICnt] = 0U;
        }
        gSBIMode = MODE_SBI_I2C_START;
        break;
```

I2C バスが空いているかどうか、“SLAVE_ADDR” データを SBI_SetSendData()に設定します。そして、送信方向を“SBI_I2C_SEND”から SBI データバッファへ設定します。その後、SBI_GenerateI2CStart(TSB_SBI0) を使用して、I2C 動作を開始します。

```
        /* Check I2C bus state and start TRx */
    case MODE_SBI_I2C_START:
        i2c_state = SBI_GetI2CState(TSB_SBI0);
        if (!i2c_state.Bit.BusState) {
            SBI_SetSendData(TSB_SBI0, SLAVE_ADDR | SBI_I2C_SEND);
            SBI_GenerateI2CStart(TSB_SBI0);
            gSBIMode = MODE_SBI_I2C_TRX;
        } else {
            /* Do nothing */
        }
        break;
```

データ転送は INTSBI0 にて処理し、データ受信は INTSBI1 にて処理します。

INTSBI0 ハンドラ内で、I2C バス状態を取得し、その値で I2C マスタ送信プロセスを決定します。I2C マスタ送信中は、SBI_SetSendData() で次のデータを送信し、I2C プロセス終了時には、SBI_GenerateI2CStop() で I2C を停止します。

```
void INTSBI0_IRQHandler(void)
{
    TSB_SBI_TypeDef *SBIx;
    SBI_I2CState sbi_sr;

    SBIx = TSB_SBI0;
    sbi_sr = SBI_GetI2CState(SBIx);
```

```
    if (sbi_sr.Bit.MasterSlave) {           /* Master mode */
        if (sbi_sr.Bit.TRx) {     /* Tx mode */
            if (sbi_sr.Bit.LastRxBit) { /* LRB=1: the receiver requires no further data. */
                SBI_GenerateI2CStop(SBIx);
            } else {                /* LRB=0: the receiver requires further data. */
                if (gI2CWCnt <= gI2CTxDataLen) {
                    SBI_SetSendData(SBIx, gI2CTxData[gI2CWCnt]);
                                                                    /* Send next data */
                    gI2CWCnt++;
                } else {            /* I2C data send finished. */
                    SBI_GenerateI2CStop(SBIx);    /* Stop I2C */
                }
            }
        } else {                    /* Rx Mode */
            /* Do nothing */
        }
    } else {                        /* Slave mode */
        /* Do nothing */
    }
}
```

INTSBI1 ハンドラで、I2C バス状態を取得し、その値で I2C スレーブ受信プロセスを決定します。SBI バッファの受信データ読み出しは SBI_GetReceiveData() 関数を用いて行います。I2C 停止状態はマスタによって制御します。

```
void INTSBI1_IRQHandler(void)
{
    uint32_t tmp = 0U;
    TSB_SBI_TypeDef *SBIy;
    SBI_I2CState sbi1_sr;

    SBIy = TSB_SBI1;
    sbi1_sr = SBI_GetI2CState(SBIy);

    if (!sbi1_sr.Bit.MasterSlave) {         /* Slave mode */
        if (!sbi1_sr.Bit.TRx) { /* Rx Mode */
            if (sbi1_sr.Bit.SlaveAddrMatch) {
                /* First read is dummy read for Slave address recognize */
                tmp = SBI_GetReceiveData(SBIy);
                gI2CRCnt = 0U;
            } else {
                /* Read I2C received data and save to I2C_RxData buffer */
                tmp = SBI_GetReceiveData(SBIy);
                gI2CRxData[gI2CRCnt] = tmp;
                gI2CRCnt++;
            }
        } else {                    /* Tx Mode */
            /* Do nothing */
        }
    } else {                        /* Master mode */
        /* Do nothing */
    }
}
```

7-11 SIO/UART

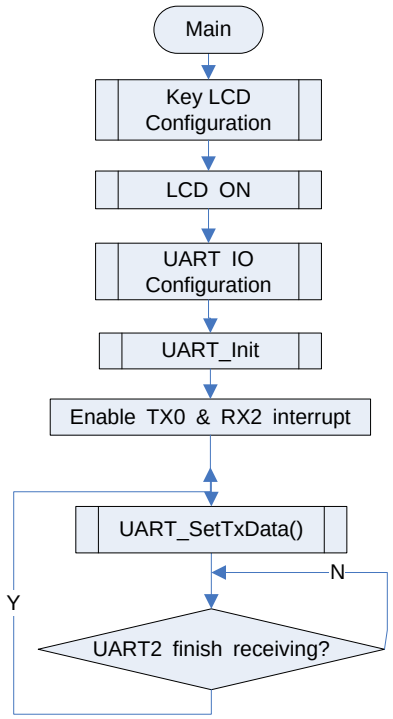
7-11-1 例: UART0/UART2 間の転送

TX03 ペリフェラルドライバ (UART, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます:

1. UART 設定と初期化
2. UART 送信制御
3. データ送信に UART0 の TX 割り込みを使用
4. UART0 からのデータ受信に UART2 の RX 割り込みを使用

- **フローチャート**



- **サンプルプログラムのコードと説明**

まず、GPIO 設定と UART の初期化を行います。
GPIO ペリフェラルドライバを使い、GPIO を UART0 に設定します。

```
GPIO_SetOutputEnableReg(GPIO_PE, GPIO_BIT_0, ENABLE);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_0);
GPIO_EnableFuncReg(GPIO_PE, GPIO_FUNC_REG_1, GPIO_BIT_1);
GPIO_SetInputEnableReg(GPIO_PE, GPIO_BIT_1, ENABLE);
```

GPIO を UART2 に設定します。

```
GPIO_SetOutputEnableReg(GPIO_PF, GPIO_BIT_0, ENABLE);
GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_1, GPIO_BIT_0);
```

```
    GPIO_EnableFuncReg(GPIO_PF, GPIO_FUNC_REG_1, GPIO_BIT_1);
    GPIO_SetInputEnableReg(GPIO_PF, GPIO_BIT_1, ENABLE);
```

UART_InitTypeDef 構造体を準備し、データを設定します。以下は設定例です。

```
    UART_InitTypeDef myUART;

    myUART.BaudRate = 115200U;    /* baut rate = 115200 */
    myUART.DataBits = UART_DATA_BITS_8; /* no handshake, 8-bit data, clock by
baud rate generator */
    myUART.StopBits = UART_STOP_BITS_1; /* 1-bit stop, LSB,    W-buff enable */
    myUART.Parity = UART_NO_PARITY;
    myUART.Mode = UART_ENABLE_RX | UART_ENABLE_TX;
    myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
```

UART のチャネル 0 とチャネル 2 を許可します。

```
  UART_Enable(UART0);
  UART_Init(UART0, &myUART);

  UART_Enable (UART2);
  UART_Init(UART2, &myUART);
```

上記設定を行い、その後 UART0 の送信割り込みと UART2 の受信割り込みを許可します。

```
  NVIC_EnableIRQ(INTTX0_IRQn);
  NVIC_EnableIRQ(INTRX2_IRQn);
```

データ送信を開始します。TxBuf は文字列配列です。

```
  UART_SetTxData(UART0, TxBuf[0]);
```

データフローの残りのプロセスは UART0 送信割り込みルーチンと UART2 の受信割り込みルーチンの ISR にて終了します。
UART0 の送信割り込みルーチン:

```
void INTTX0_IRQHandler(void)
{
    volatile UART_Err err;

    if (TxCounter < NumToBeTx) {
        UART_SetTxData(UART0, TxBuffer[TxCounter++]);
    } else {
        err = UART_GetErrState(UART0);
    }
}
```

UART2 受信割り込みルーチン:

```
void INTRX2_IRQHandler(void)
{
    volatile UART_Err err;

    err = UART_GetErrState(UART2);
    if (UART_NO_ERR == err) {
        RxBuffer[RxCounter++] = (uint8_t) UART_GetRxData(UART2);
    }
}
```

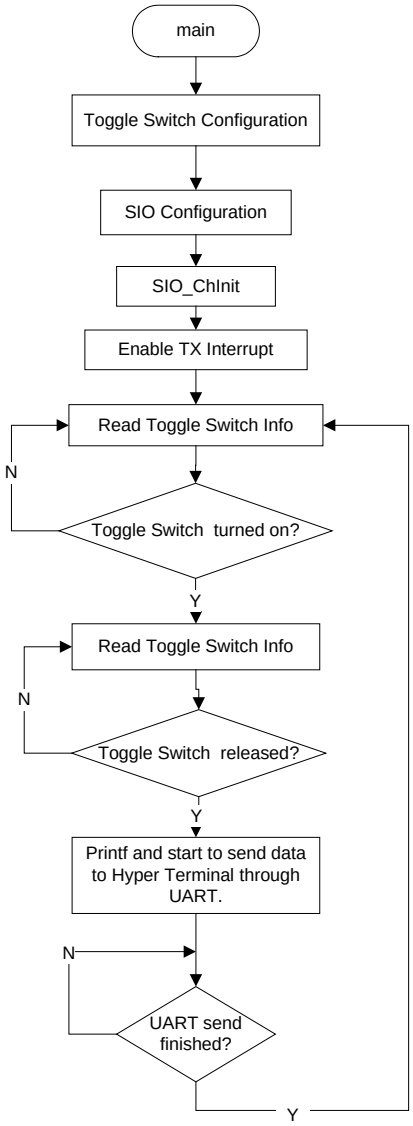
7-11-2　例: リターゲット

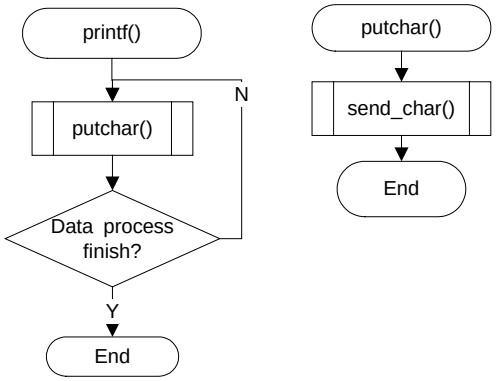
TX03 ペリフェラルドライバ(UART、GPIO)を使用した簡単なサンプルプログラムです。

以下の例が含まれます:

1. UART 設定と初期化
2. UART 送信制御
3. データ送信に UART の TX 割り込みを使用
4. UART に printf()関数をリターゲット

- **フローチャート**





- **サンプルプログラムのコードと説明**

まず、GPIO を設定し、UART を初期化します。

```
    GPIO_SetOutputEnableReg(GPIO_PC, GPIO_BIT_0, ENABLE);
    GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_0);
    GPIO_EnableFuncReg(GPIO_PC, GPIO_FUNC_REG_1, GPIO_BIT_1);
    GPIO_SetInputEnableReg(GPIO_PC, GPIO_BIT_1, ENABLE);
    GPIO_SetPullUp(GPIO_PC, GPIO_BIT_0, ENABLE);
    GPIO_SetPullUp(GPIO_PC, GPIO_BIT_1, ENABLE);
```

UART_InitTypeDef 構造体を準備し、データを設定します。以下は設定例です。

```
    UART_InitTypeDef myUART;

    /* configure SIO1 for reception */
    UART_Enable(UART_RETARGET);
    myUART.BaudRate = 115200U;    /* baud rate = 115200 */
    myUART.DataBits = UART_DATA_BITS_8; /* no handshake, 8-bit data, clock by
baud rate generator */
    myUART.StopBits = UART_STOP_BITS_1; /* 1-bit stop, LSB,    W-buff enable */
    myUART.Parity = UART_NO_PARITY;
    myUART.Mode = UART_ENABLE_TX;
    myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
    UART_Init(UART_RETARGET, &myUART);
```

上記設定を行い、その後、UART の送信割り込みを有効にします。

```
    NVIC_EnableIRQ(RETARGET_INT)
```

その後データ送信を開始します。ここでの TxBuffer は文字列です。

```
    printf("%s\r\n", TxBuffer);
```

データフローの残りのプロセスは UART1 送信割り込みルーチンの ISR にて終了します。
UART0 の送信割り込みルーチン:

```
void INTTX1_IRQHandler(void)
{
    if (gSIORdIndex < gSIOWrIndex) {       /* buffer is not empty */
        UART_SetTxData(UART_RETARGET,          gSIOTxBuffer[gSIORdIndex++]);
/* send data */
        fSIO_INT = SET;              /* SIO1 INT is enable */
    } else {
        /* disable SIO1 INT */
        fSIO_INT = CLEAR;
        NVIC_DisableIRQ(RETARGET_INT);
        fSIOTxOK = YES;
```

```
    }
    if (gSIORdIndex >= gSIOWrIndex) {    /* reset buffer index */
        gSIOWrIndex = CLEAR;
        gSIORdIndex = CLEAR;
    } else {
        /* Do nothing */
    }
}
```

printf()関数は IAR コンパイラの putchar()を、RealView コンパイラの fputc()をコールしてデータ出力を行います。

```
#if defined ( __CC_ARM    )          /* RealView Compiler */
struct __FILE {
    int handle;                      /* Add whatever you need here */
};
FILE __stdout;
FILE __stdin;
int fputc(int ch, FILE * f)
#elif defined ( __ICCARM__ )         /*IAR Compiler */
int putchar(int ch)
#endif
{
    return (send_char(ch));
}

uint8_t send_char(uint8_t ch)
{
    while (gSIORdIndex != gSIOWrIndex) {          /* wait for finishing sending */
        /* do nothing */
    }
    gSIOTxBuffer[gSIOWrIndex++] = ch;    /* fill TxBuffer */
    if (fSIO_INT == CLEAR) {     /* if SIO INT disable, enable it */
        fSIO_INT = SET;              /* set SIO INT flag */
        UART_SetTxData(UART_RETARGET, gSIOTxBuffer[gSIORdIndex++]);
        NVIC_EnableIRQ(RETARGET_INT);
    }

    return ch;
}
```

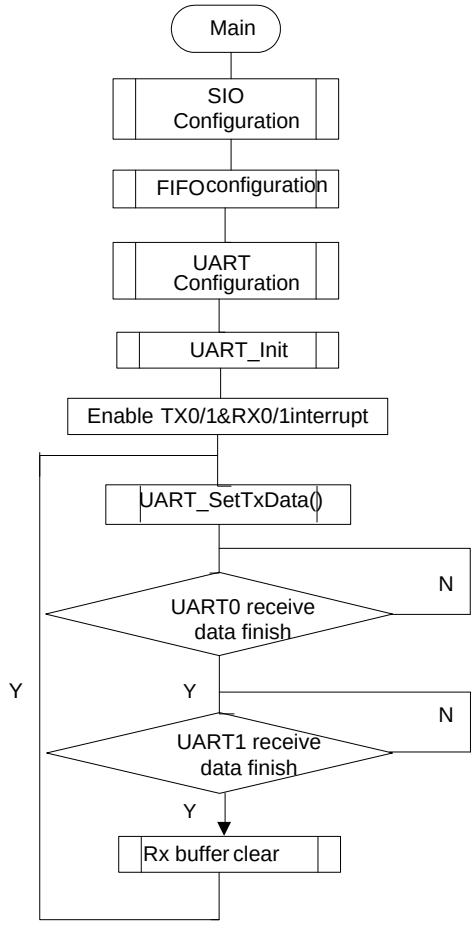
7-11-3　例: UART FIFO

UART と GPIO を使用した簡単なサンプルプログラムです。

以下の例が含まれます:

1. UART と FIFO の初期設定
2. FIFO を使用した UART の送受信

- **フローチャート**



- **サンプルプログラムのコードと説明**

最初に GPIO の設定と UART の初期化を行います。

GPIO ドライバを使用して、GPIO を UART0 と UART1 に設定します。

```
void SIO_Configuration(TSB_SC_TypeDef * SCx)
{
	if (SCx == TSB_SC0) {
		TSB_PE->CR |= GPIO_BIT_0;
		TSB_PE->FR1 |= GPIO_BIT_0;
		TSB_PE->FR1 |= GPIO_BIT_1;
		TSB_PE->IE |= GPIO_BIT_1;
	} else if (SCx == TSB_SC1) {
		TSB_PC->CR |= GPIO_BIT_0;
		TSB_PC->FR1 |= GPIO_BIT_0;
		TSB_PC->FR1 |= GPIO_BIT_1;
		TSB_PC->IE |= GPIO_BIT_1;
	}
}
```

UART_InitTypeDef 構造体を用意し、すべてのメンバを設定します。以下は設定例です。

```
	UART_InitTypeDef myUART;

   /* configure SIO0 for reception */
```

```
    UART_Enable(UART_RETARGET);
    myUART.BaudRate = 115200U;    /* baud rate = 115200 */
    myUART.DataBits = UART_DATA_BITS_8; /* no handshake, 8-bit data, clock by baud rate generator */
    myUART.StopBits = UART_STOP_BITS_1; /* 1-bit stop, LSB,    W-buff enable */
    myUART.Parity = UART_NO_PARITY;
    myUART.Mode = UART_ENABLE_TX|UART_ENABLE_RX;
    myUART.FlowCtrl = UART_NONE_FLOW_CTRL;
    UART_Init(UART_RETARGET, &myUART);
```

UART のドライバを使用して、UART0/1 の各チャネルの許可と初期設定を行います。

```
    UART_Enable(UART0);
    UART_Init(UART0, &myUART);

    UART_Enable(UART1);
    UART_Init(UART1, &myUART);
```

FIFO の設定を行います。

```
    UART_RxFIFOByteSel(UART0,UART_RXFIFO_RXFLEVEL);
    UART_RxFIFOByteSel(UART1,UART_RXFIFO_RXFLEVEL);

    UART_TxFIFOINTCtrl(UART0,ENABLE);
    UART_TxFIFOINTCtrl(UART1,ENABLE);

    UART_RxFIFOINTCtrl(UART0,ENABLE);
    UART_RxFIFOINTCtrl(UART1,ENABLE);

    UART_TRxAutoDisable(UART0,UART_RXTXCNT_AUTODISABLE);
    UART_TRxAutoDisable(UART1,UART_RXTXCNT_AUTODISABLE);

    UART_FIFOConfig(UART0,ENABLE);
    UART_FIFOConfig(UART1,ENABLE);

    UART_RxFIFOFillLevel(UART0, UART_RXFIFO4B_FLEVLE_4_2B);
    UART_RxFIFOFillLevel(UART1, UART_RXFIFO4B_FLEVLE_4_2B);

    UART_RxFIFOINTSel(UART0,UART_RFIS_REACH_EXCEED_FLEVEL);
    UART_RxFIFOINTSel(UART1,UART_RFIS_REACH_EXCEED_FLEVEL);

    UART_RxFIFOClear(UART0);
    UART_RxFIFOClear(UART1);

    UART_TxFIFOFillLevel(UART0, UART_TXFIFO4B_FLEVLE_0_0B);
    UART_TxFIFOFillLevel(UART1, UART_TXFIFO4B_FLEVLE_0_0B);

    UART_TxFIFOINTSel(UART0,UART_TFIS_REACH_NOREACH_FLEVEL);
    UART_TxFIFOINTSel(UART1,UART_TFIS_REACH_NOREACH_FLEVEL);

    UART_TxFIFOClear(UART0);
    UART_TxFIFOClear(UART1);
```

上記設定を行い、その後 UART0/1 の各割り込みを許可します。

```
    NVIC_EnableIRQ(INTTX0_IRQn);
    NVIC_EnableIRQ(INTRX1_IRQn);

    NVIC_EnableIRQ(INTTX1_IRQn);
    NVIC_EnableIRQ(INTRX0_IRQn);
```

最後に UART0/1 の送信割り込みと受信割り込みの割り込み処理ルーチンを準備します。
UART0 の送信割り込み処理ルーチン:

```
void INTTX0_IRQHandler(void)
{
	volatile UART_Err err;

	if (TxCounter < NumToBeTx) {
		UART_SetTxData(UART0, TxBuffer[TxCounter++]);
	} else {
		err = UART_GetErrState(UART0);
	}
}
```

UART1 の送信割り込み処理ルーチン:

```
void INTTX1_IRQHandler(void)
{
	volatile UART_Err err;

	if (TxCounter1 < NumToBeTx1) {
		UART_SetTxData(UART1, TxBuffer1[TxCounter1++]);
	} else {
		err = UART_GetErrState(UART1);
	}
}
```

UART1 の受信割り込み処理ルーチン:

```
void INTRX0_IRQHandler(void)
{
	volatile UART_Err err;

	err = UART_GetErrState(UART0);
	if (UART_NO_ERR == err) {
		RxBuffer[RxCounter++] = (uint8_t) UART_GetRxData(UART0);
	}
}
```

UART1 の受信割り込み処理ルーチン:

```
void INTRX1_IRQHandler(void)
{
	volatile UART_Err err;

	err = UART_GetErrState(UART1);
	if (UART_NO_ERR == err) {
		RxBuffer1[RxCounter1++] = (uint8_t) UART_GetRxData(UART1);
	}
}
```

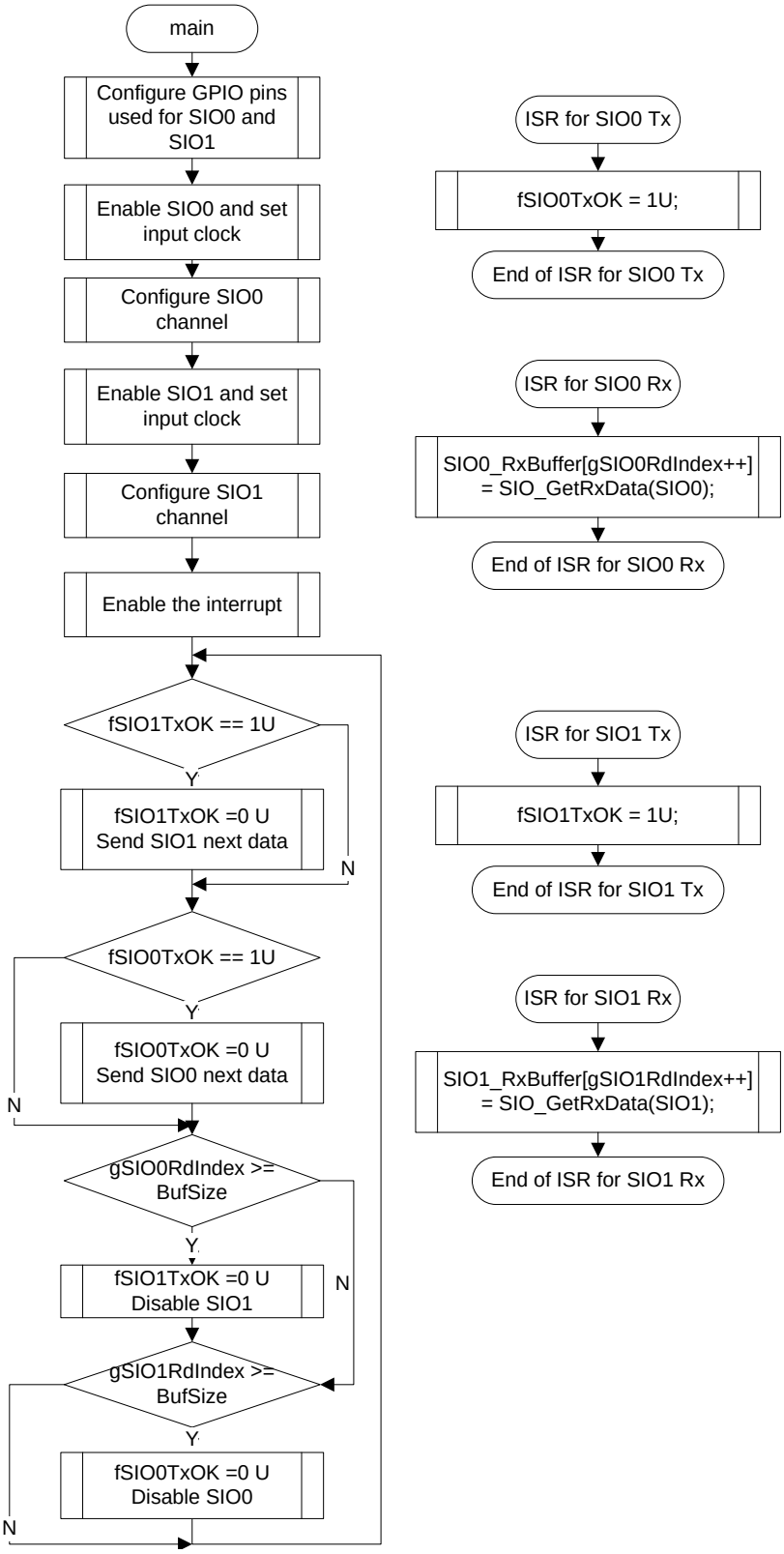
7-11-4　例: SIO

TX03 ペリフェラルドライバ (SIO, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます。

1. SIO の基本設定
2. マスタ SIO0⇔スレーブ SIO0 間の通信制御
3. 送受信に SIO 割り込みを使用

- **フローチャート**


main
Configure GPIO pins used for SIO0 and SIO1
Enable SIO0 and set input clock
Configure SIO0 channel
Enable SIO1 and set input clock
Configure SIO1 channel
Enable the interrupt
fSIO1TxOK == 1U
Y
fSIO1TxOK =0 U
Send SIO1 next data
N
fSIO0TxOK == 1U
Y
fSIO0TxOK =0 U
Send SIO0 next data
N
gSIO0RdIndex >= BufSize
Y
fSIO1TxOK =0 U
Disable SIO1
N
gSIO1RdIndex >= BufSize
Y
fSIO0TxOK =0 U
Disable SIO0
N
ISR for SIO0 Tx
fSIO0TxOK = 1U;
End of ISR for SIO0 Tx
ISR for SIO0 Rx
SIO0_RxBuffer[gSIO0RdIndex++] = SIO_GetRxData(SIO0);
End of ISR for SIO0 Rx
ISR for SIO1 Tx
fSIO1TxOK = 1U;
End of ISR for SIO1 Tx
ISR for SIO1 Rx
SIO1_RxBuffer[gSIO1RdIndex++] = SIO_GetRxData(SIO1);
End of ISR for SIO1 Rx

- **サンプルプログラムのコードと説明**

まず、GPIO 設定と SIO の初期化を行います。

GPIO ペリフェラルドライバを使い、GPIO を SIO0 に設定します。その後、SIO0 の初期化構造体を準備し、入力クロックの初期化を行います。

```
/*Enable the SIO0 channel */
SIO_Enable(SIO0);

/*initialize the SIO0 struct */
SIO0_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO0_Init.IntervalTime = SIO_SINT_TIME_SCLK_8;
SIO0_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO0_Init.TransferDir = SIO_LSB_FRIST;
SIO0_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO0_Init.DoubleBuffer = SIO_WBUF_ENABLE;
SIO0_Init.BaudRateClock = SIO_BR_CLOCK_T4;
SIO0_Init.Divider = SIO_BR_DIVIDER_2;
SIO_Init(SIO0, SIO_CLK_BAUDRATE, &SIO0_Init);
```

SIO1 の初期化構造体を準備し、入力クロックの初期化を行います。

```
/*Enable the SIO1 channel */
SIO_Enable(SIO1);

/*initialize the SIO1 struct */
SIO1_Init.InputClkEdge = SIO_SCLKS_TXDF_RXDR;
SIO1_Init.TransferMode = SIO_TRANSFER_FULLDPX;
SIO1_Init.TransferDir = SIO_LSB_FRIST;
SIO1_Init.Mode = SIO_ENABLE_TX | SIO_ENABLE_RX;
SIO1_Init.DoubleBuffer = SIO_WBUF_ENABLE;

SIO_Init(SIO1, SIO_CLK_SCLKINPUT, &SIO1_Init);
```

SIO の送受信割り込みを有効にします。

```
/* Enable SIO0 Channel TX interrupt */
NVIC_EnableIRQ(INTTX0_IRQn);
/* Enable SIO1 Channel RX interrupt */
NVIC_EnableIRQ(INTRX1_IRQn);

/* Enable SIO1 Channel TX interrupt */
NVIC_EnableIRQ(INTTX1_IRQn);
/* Enable SIO0 Channel RX interrupt */
NVIC_EnableIRQ(INTRX0_IRQn);
```

すべての基本設定を行い、その後、データ送信を行います。

```
while (1) {
        /* SIO1 send data from TXD1*/
        if (fSIO1TxOK == 1U) {
            fSIO1TxOK = 0U;
            SIO_SetTxData(SIO1, SIO1_TxBuffer[gSIO1WrIndex++]);
        } else {
            /*Do Nothing */
        }
        /* SIO0 send data from TXD0*/
        if (fSIO0TxOK == 1U) {
            fSIO0TxOK = 0U;
            SIO_SetTxData(SIO0, SIO0_TxBuffer[gSIO0WrIndex++]);
```

```
        } else {
            /*Do Nothing */
        }

        /*SIO0 receive data end */
        if (gSIO0RdIndex >= BufSize) {
            fSIO1TxOK = 0U;
            SIO_Disable(SIO1);
        } else {
            /*Do Nothing */
        }
        /*SIO1 receive data end */
        if (gSIO1RdIndex >= BufSize) {
            fSIO0TxOK = 0U;
            SIO_Disable(SIO0);
        } else {
            /*Do Nothing */
        }    }
```

SIO0 送信の ISR にて、転送設定を行います。

```
void INTTX0_IRQHandler (void)
{
    fSIO0TxOK = 1U;
}
```

SIO0 受信の ISR にて、受信バッファからデータを受信します。

```
void INTRX0_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO0RdIndex++] = SIO_GetRxData(SIO0);
}
```

SIO1 送信の ISR にて、転送設定を行います。

```
void INTTX1_IRQHandler(void)
{
    fSIO1TxOK = 1U;
}
```

SIO1 受信の ISR にて、受信バッファからデータを受信します。

```
void INTRX1_IRQHandler(void)
{
    SIO0_RxBuffer[gSIO1RdIndex++] = SIO_GetRxData(SIO1);
}
```

7-12 SSP

7-12-1 例: SSP0 から SSP1 への DMAC 転送

TX03 ペリフェラルドライバ(SSP, DMAC, GPIO)を使用したサンプルソフトです。

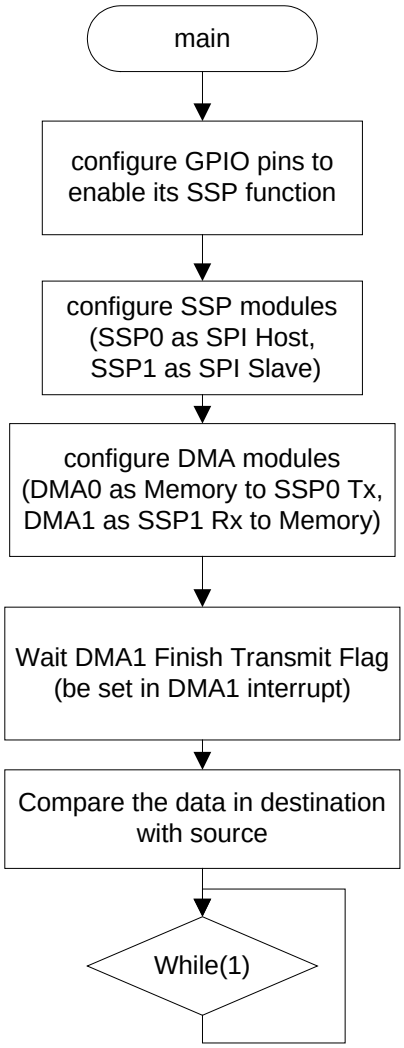
以下の例を含みます。

1. SSP0 を SPI ホストとする設定と、SPI スレーブとする設定
2. DMAC0 をメモリから周辺回路(SSP0 TX)とする設定と DMAC1 を周辺回路(SSP1 RX)

からメモリとする設定

3. データ転送フローの確認: メモリ→SSP0 TX→SSP1 RX→メモリ

- **フローチャート**



- **サンプルプログラムのコードと説明**

以下は上記フローチャートの main.c です。

```
int main(void)
{
	GPIO_SetSSP();
	initSSP();
	InitDMA();

	/* Wait the end of transmission */
	while (TxEndFlag != DONE) {
		/* Do nothing */
	}

	/* now DMA is finished, Set a Break Point here, */
	/* after function Buffercompare() is called, result == SAME */
	result = Buffercompare( SRC_Buffer, DST_Buffer, BUFFER_SIZE);
```

```
    while(1) {
        /* do nothing */
    }
}
```

上記関数 GPIO_SetSSP(), initSSP(), InitDMA()の詳細は、コードのコメントを参照してください。

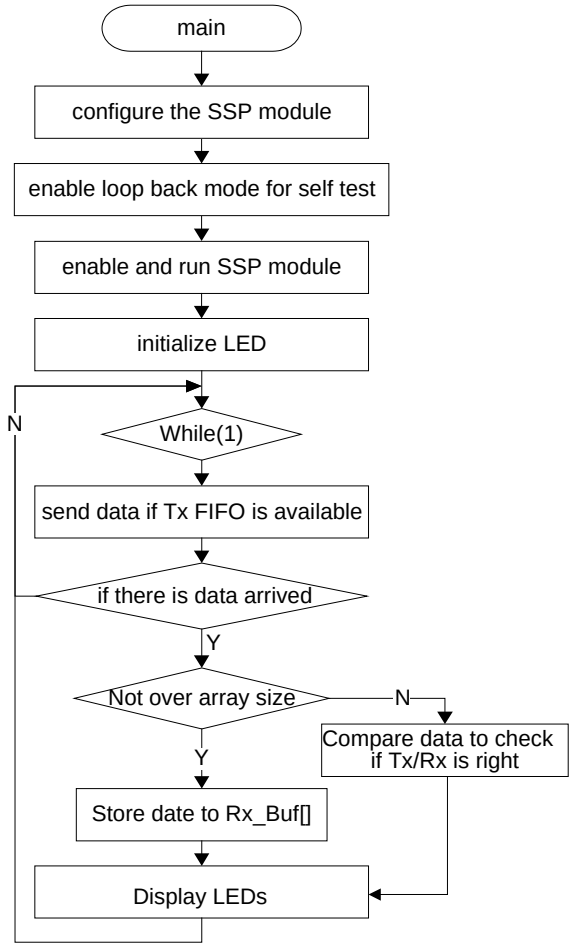
7-12-2 例: SSP0 セルフループバック

TX03 ペリフェラルドライバ(SSP, GPIO)を使用したサンプルソフトです。

以下の例を含みます。

1. SSP0 の設定と初期化
2. 自身で送受信を行うループバックモードの許可

- **フローチャート**



- **サンプルプログラムのコードと説明**

```
/* main function */
int main(void)
{
	SSP_InitTypeDef initSSP;
	SSP_FIFOState fifoState;

	uint16_t datTx = 0U;		/* must use 16bit type */
	uint32_t cntTx = 0U;
	uint32_t cntRx = 0U;

	uint16_t receive = 0U;
	uint16_t Rx_Buf[MAX_BUFSIZE] = { 0U };
	uint16_t Tx_Buf[MAX_BUFSIZE] = { 0U };

	/* configure the SSP module */
	initSSP.FrameFormat = SSP_FORMAT_SPI;

	/* default is to run at maximum bit rate */
	initSSP.PreScale = 2U;
	initSSP.ClkRate = 1U;
	/* define BITRATE_MIN to run at minimum bit rate */
	/*	BitRate = fSYS / (PreScale x (1 + ClkRate)) */
#ifdef BITRATE_MIN
	initSSP.PreScale = 254U;
	initSSP.ClkRate = 255U;
#endif
	initSSP.ClkPolarity = SSP_POLARITY_LOW;
	initSSP.ClkPhase = SSP_PHASE_FIRST_EDGE;
	initSSP.DataSize = 16U;
	initSSP.Mode = SSP_MASTER;
	SSP_Init(TSB_SSP0, &initSSP);

	/* enable loop back mode for self test */
	SSP_SetLoopBackMode(TSB_SSP0, ENABLE);

	/* enable and run SSP module */
	SSP_Enable(TSB_SSP0);

	/* initialize LEDs on M384-SK board before display something */
	LED_Init();

	while (1) {

		datTx++;
		/* send data if Tx FIFO is available */
		fifoState = SSP_GetFIFOState(TSB_SSP0, SSP_TX);
		if ((fifoState == SSP_FIFO_EMPTY) || (fifoState == SSP_FIFO_NORMAL)) {
			SSP_SetTxData(TSB_SSP0, datTx);
			if (cntTx < MAX_BUFSIZE) {
				Tx_Buf[cntTx] = datTx;
				cntTx++;
			} else {
				/* do nothing */
			}
		} else {
			/* do nothing */
```

```
            }

            /* check if there is data arrived */
            fifoState = SSP_GetFIFOState(TSB_SSP0, SSP_RX);
            if ((fifoState == SSP_FIFO_FULL) || (fifoState == SSP_FIFO_NORMAL)) {
                  receive = SSP_GetRxData(TSB_SSP0);
                  if (cntRx < MAX_BUFSIZE) {
                        Rx_Buf[cntRx] = receive;
                        cntRx++;
                  } else {
                   /* Place a break point here to check if receive data is right.            */
                   /* Success Criteria:                                                      */
                   /*   Every data transmited from Tx_Buf is received in Rx_Buf.         */
                   /*   When the line "#define BITRATE_MIN" is commented, the SSP is run in maxium */
                   /*   bit rate, so we can find there is enough time to transmit date from 1 to */
                   /*   MAX_BUFSIZE one by one. but if we uncomment that line, SSP is run in */
                   /*   minimum bit rate, we will find that receive data can't catch "datTx++", */
                   /*   in this so slow bit rate, when the Tx FIFO is available, the cntTx has */
                   /*   been increased so much.                                              */
                        __NOP();
                        result = Buffercompare( Tx_Buf, Rx_Buf, MAX_BUFSIZE);

                        if (result == NOT_SAME) {
                              LED_Off(LED1);
                              LED_On(LED0);
                        } else {
                              LED_On(LED1);
                              LED_Off(LED0);
                        }
                  }

            } else {
                  /* do nothing */
            }

            DisplayLED(receive);
      }
}
```

7-13 TMRB

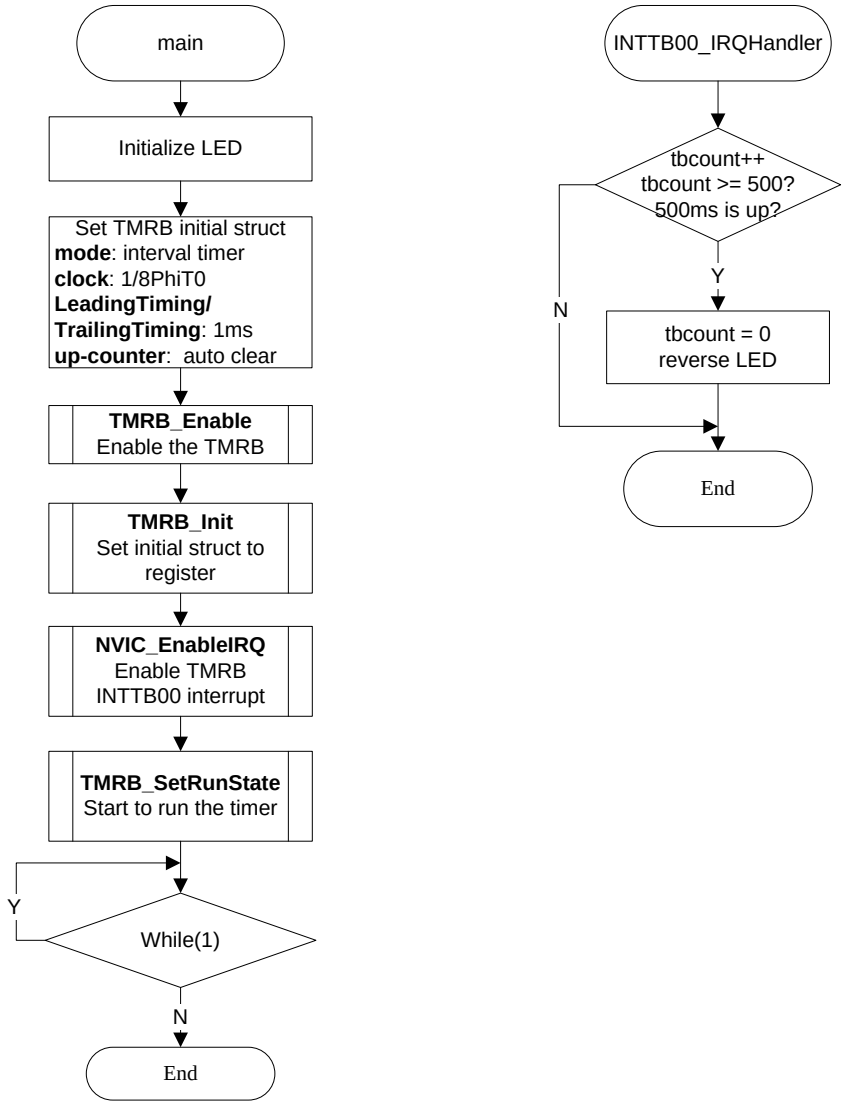
7-13-1 例: 汎用タイマ

TX03 ペリフェラルドライバ(TMRB, GPIO)を使用した簡単なサンプルプログラムです。

以下の例が含まれます:

1. TMRB0 の初期化
2. 1ms の汎用タイマ

- **フローチャート**



- **サンプルプログラムのコードと説明**

最初に LED を初期化し、LED を点灯します。

```
LED_Init();                                    /* LED initialize */
LED_On(LED_ALL);                               /* Turn on LED_ALL */
```

TMRB 設定用の構造体を用意し TMRB モード、クロック、アップカウンタクリア方法、周期とデューティを設定します。このサンプルでは、1ms の周期とデューティを設定します。TMRB_1MS マクロは、0x1388 です。(φT0=fsys=10MHz * PLL* = 40MHz, ftmrb = 1/8φT0 = 5MHz, Ttmrb = 0.2us, 1ms/0.2us = 5000 = 0x1388 (クロック設定についての詳細は CG 章を参照してください))

```
TMRB_InitTypeDef m_tmrb;
```

```
m_tmrb.Mode = TMRB_INTERVAL_TIMER;          /* internal timer */
m_tmrb.ClkDiv = TMRB_CLK_DIV_8;             /* 1/8PhiT0 */
m_tmrb.TrailingTiming = TMRB_1MS;           /* periodic time is 1ms */
m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;         /* up-counter auto clear */
m_tmrb.LeadingTiming = TMRB_1MS;            /* periodic time is 1ms */
```

TMRB モジュールを有効にした後、初期化構造体を指定のレジスタに設定します。INTTB00 割り込み（1ms ごとにトリガ）を有効にします。最後に TMRB を動作させます。

```
TMRB_Enable(TSB_TB0);                       /* enable the TMRB0 */
TMRB_Init(TSB_TB0, &m_tmrb);                /* initial the TMRB0 */
NVIC_EnableIRQ(INTTB0_IRQn);                /* enable INTTB0 interrupt */
TMRB_SetRunState(TSB_TB0, TMRB_RUN);        /* run TMRB0*/
```

メインルーチンは、"While(1) "に入り、割り込みの発生を待ちます。割り込みルーチンでは、カウンタでカウントを行い、500ms をカウントすると、 LED を反転させカウントを再開します。

```
tbcount++;
if (tbcount >= 500U) {                      /* 500ms is up */
    tbcount = 0U;
    /* reverse LED output */
    ledon = (ledon == 0U) ? 1U : 0U;
    if (0U == ledon) {
        LED_Off(LED_ALL);
    } else {
        LED_On(LED_ALL);
    }
} else {
    /* do nothing */
}
```

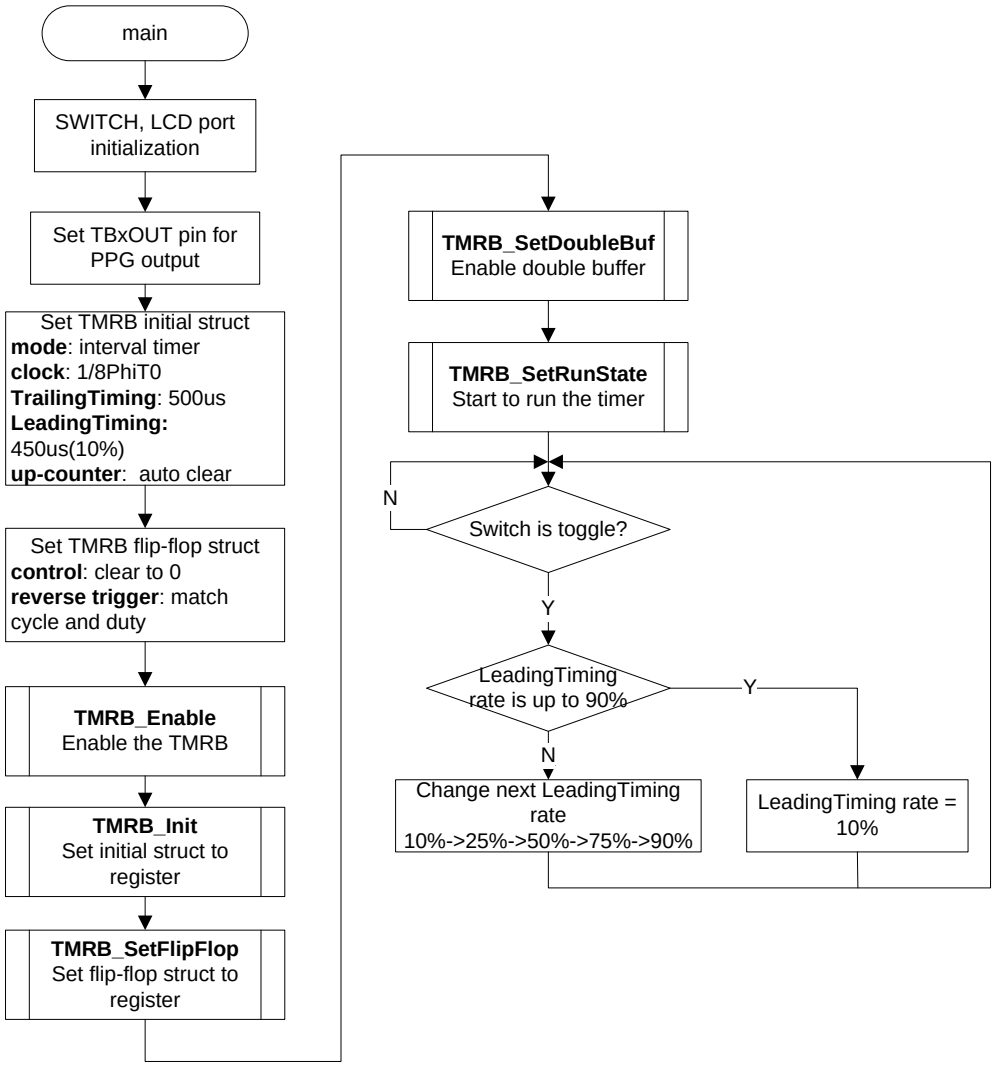
7-13-2 例: PPG 波形出力

TX03 ペリフェラルドライバ(TMRB, GPIO)を使用した簡単なサンプルプログラムです。

以下の例が含まれます:

1. TMRB6 の初期化
2. PPG 機能の設定と開始
3. PPG デューティの調整

- **フローチャート**



- **サンプルプログラムのコードと説明**

まず、LCD とスイッチの初期化を行い、PPG 出力用に PK1 を TB6OUT に設定します。

```
/* LCD & switch initialization */
    LCD_Configuration();
    SW_Init();

    /* Set PA5 as TB6OUT for PPG output */
    GPIO_SetOutput(GPIO_PA, GPIO_BIT_5);
    GPIO_EnableFuncReg(GPIO_PA, GPIO_FUNC_REG_2, GPIO_BIT_5);
```

TMRB の初期化用構造体を準備し、TMRB モード、クロック、アップカウンタクリア設定、サイクル、デューティを設定します。この例では 500us のサイクルを設定するため、TMRB6TIME マクロを定義してあります。このマクロは 0x09C4 です。(φT0 = fsys = fc = 10MHz * PLL * = 40MHz, ftmrb = 1/8 φT0 = 5MHz, Ttmrb = 0.2us, 500us/0.2us = 2500 = 0x09C4 (クロック設定の詳細は CG 章を参照してください))

```
    TMRB_InitTypeDef m_tmrb;

    m_tmrb.Mode = TMRB_INTERVAL_TIMER;          /* internal timer */
    m_tmrb.ClkDiv = TMRB_CLK_DIV_8;             /* 1/8PhiT0 */
    m_tmrb.TrailingTiming = TMRB6TIME;          /* trailing timing is 500us */
    m_tmrb.UpCntCtrl = TMRB_AUTO_CLEAR;         /* up-counter auto clear */
    m_tmrb.LeadingTiming = LeadingTiming[Rate];     /* leading timing, initial value
10% */
```

フリッププロップ初期化構造体を用意し、フリップフロップ制御、反転トリガ引数を設定します。反転トリガは、デューティーとサイクルを一致するように設定します。

```
    PPGFFInital.FlipflopCtrl = TMRB_FLIPFLOP_CLEAR;
    PPGFFInital.FlipflopReverseTrg=TMRB_FLIPFLOP_MATCH_TRAILINGTIMING|
    TMRB_FLIPFLOP_MATCH_LEADINGTIMING;
```

TMRB モジュールを許可し、指定レジスタに初期化構造体、フリップフロップ構造体を設定します。ダブルバッファを許可し、キャプチャ機能を禁止にします。最後に、TMRB を動作させます。

```
    TMRB_Enable(TSB_TB6);
    TMRB_Init(TSB_TB6, &m_tmrb);
    TMRB_SetFlipFlop(TSB_TB6, &PPGFFInital);
    /* enable double buffer */
    TMRB_SetDoubleBuf(TSB_TB6,ENABLE, TMRB_WRITE_REG_SEPARATE);
    TMRB_SetRunState(TSB_TB6, TMRB_RUN);
```

スイッチ状態の変化を待ち、現在のデューティ状態を LED に表示します。

```
    do {                            /* wait if switch is Low */
            keyvalue = GPIO_ReadDataBit(KEYPORT, GPIO_BIT_0);
            LeadingTiming_display(); /* display current leading timing */
       } while (GPIO_BIT_VALUE_0 == keyvalue);
```

スイッチ状態が変化すると、下記のようにデューティーを設定します。
10%->25%->50%->75% ->90%その後 再び 90%から 10%になります。

```
    Rate++;
    if (Rate >= LEADINGTIMINGMAX) {
        Rate = LEADINGTIMINGINIT;
        } else {
            /* Do nothing */
        }

    TMRB_ChangeLeadingTiming(TSB_TB6,   LeadingTiming[Rate]);       /*  change
leading timing rate */
```

デューティの算出方法:
TrailingTiming = 500us, ftmrb = 1/8 fphiT0 = 5MHz, Ttmrb = 0.2us (これらの時間に関するパラメータは、CG 設定により異なります)

LeadingTiming = 10%: High 幅は 500*10% = 50us, Low 幅は 500-50 = 450us, カウンタ値 = 450us/Ttmrb = 0x8CAU

LeadingTiming = 25%: High 幅は 500*25% = 125us, Low 幅は 500-125 = 375us, カウン

タ値= 375us/Ttmrb = 0x753U

LeadingTiming = 50%: High 幅は 500*50% = 250us, Low 幅は 500-250 = 250us, カウンタ値= 250us/Ttmrb = 0x4E2

LeadingTiming = 75%: High 幅は 500*75% = 375us, Low 幅は 500-375 = 125us, カウンタ値 = 125us/Ttmrb = 0x271

LeadingTiming = 90%: High 幅は 500*90% = 450us, Low 幅は 500-450 = 50us, カウンタ値 = 50us/Ttmrb = 0xFA

これは上記計算式から求めたデューティ値の配列です。

```
uint32_t LeadingTiming[5] = { 0x8CAU, 0x753U, 0x4E2U, 0x271U, 0xFAU };
/* leading timing: 10%, 25%, 50%, 75%, 90% */
```

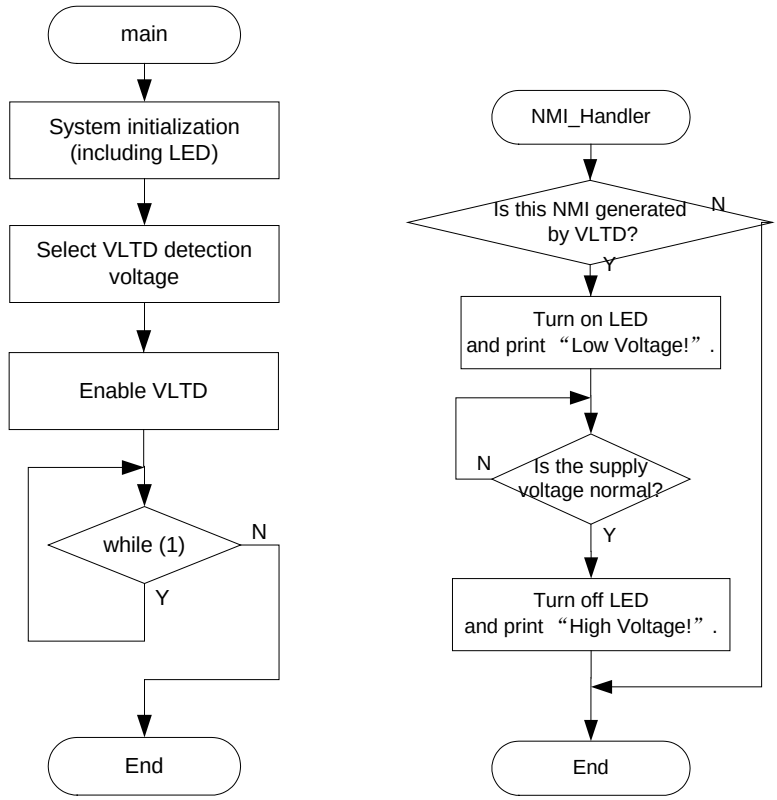
7-14 VLTD

TX03 ペリフェラルドライバ(VLTD, GPIO)を用いたサンプルプログラムです。

この例では以下を行います。

1. VLTD の初期化。
2. VLTD 状態の初期化。

- **フローチャート**



- **サンプルプログラムのコードと説明**

最初にシステムを初期化します。不用意な WDT 割割り込みを避けるため WDT を無効にします。

```
WDT_Disable ();
LEDInit();
```

検出電圧の設定と VLTD 動作を許可します。

```
VLTD_SetVoltage(VLTD_DETECT_VOLTAGE_46);
VLTD_Enable ();
```

その後、while(1)ループさせます。
VLTD が電源電圧の低下を検出し NMI の発生を検出すると、LED を点灯し、ターミナルウィンドウにメッセージを表示します。

```
if (CG_GetNMIFlag().Bit.VoltageDetection == 1U) {
{
        LedOn(LED3);
#ifdef DEBUG
        printf("Low voltage!\n");
#endif
```

その後、電源電圧が通常に戻ると、LED を消灯し、ターミナルウィンドウにメッセージを表示します。

```
        while (VLTD_GetStatus() == 1U)
        {
            __NOP();
```

```
        }
        LedOff(LED3);
#ifdef DEBUG
        printf("High voltage!\n");
#endif
}
```

最後に NMI_Handler()関数に戻ります。

7-15 WDT

TX03 ペリフェラルドライバ (WDT, GPIO) のプログラムサンプルです。

この例では以下を行います。

1. WDT の初期化。
2. DEMO1 では、オーバーフロー前に WDT クリアを行わず、NMI 割り込みを発生させます。
3. DEMO2 では、オーバーフロー前に WDT クリアを行い、常時 LED0 を点滅させます。

- **サンプルプログラムのコードと説明**

以下のコードはWDT の初期化の例です。検出時間が 2^25/fsys にされ、オーバーフロー時に NMI 割り込みを発生します。

```
WDT_InitTypeDef WDT_InitStruct;
WDT_InitStruct.DetectTime = WDT_DETECT_TIME_EXP_25;
WDT_InitStruct.OverflowOutput = WDT_NMIINT;
```

WDT を初期化し、その後 WDT を有効にします。

```
WDT_Init(&WDT_InitStruct);
WDT_Enable();
```

DEMO1 では、NMI 割り込みの発生を待ちます。

```
while(1)
{
}
```

DEMO1 では、NMI 割り込み発生時に WDT を禁止にし、LED1 の点滅を停止します。

```
WDT_Disable();
```

DEMO2 では、WDT クリアを行い、常に LED0 を点滅させます。

```
WDT_WriteClearCode();
```

7-16 ENC

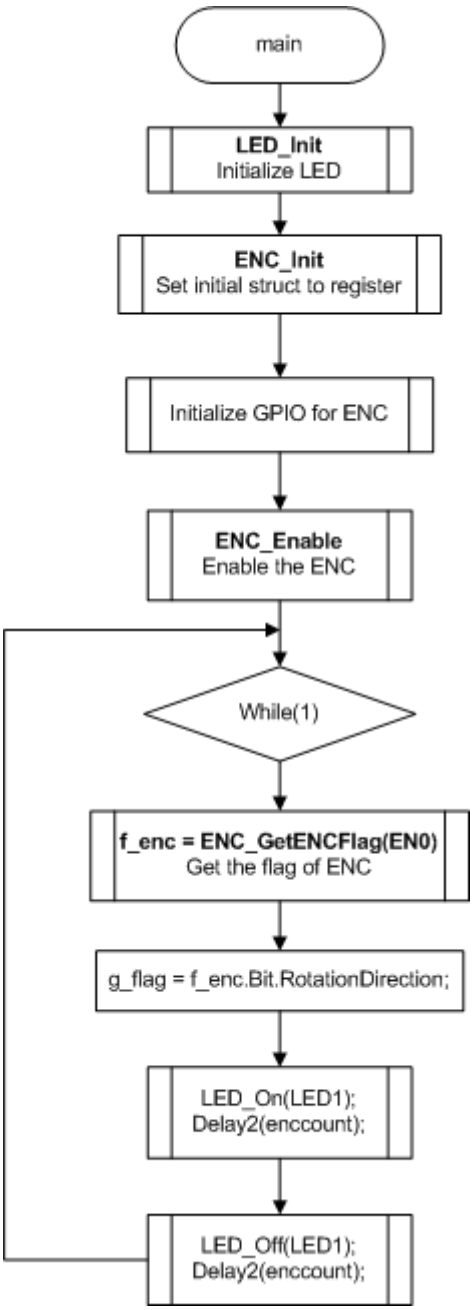
7-16-1 例: 回転検出

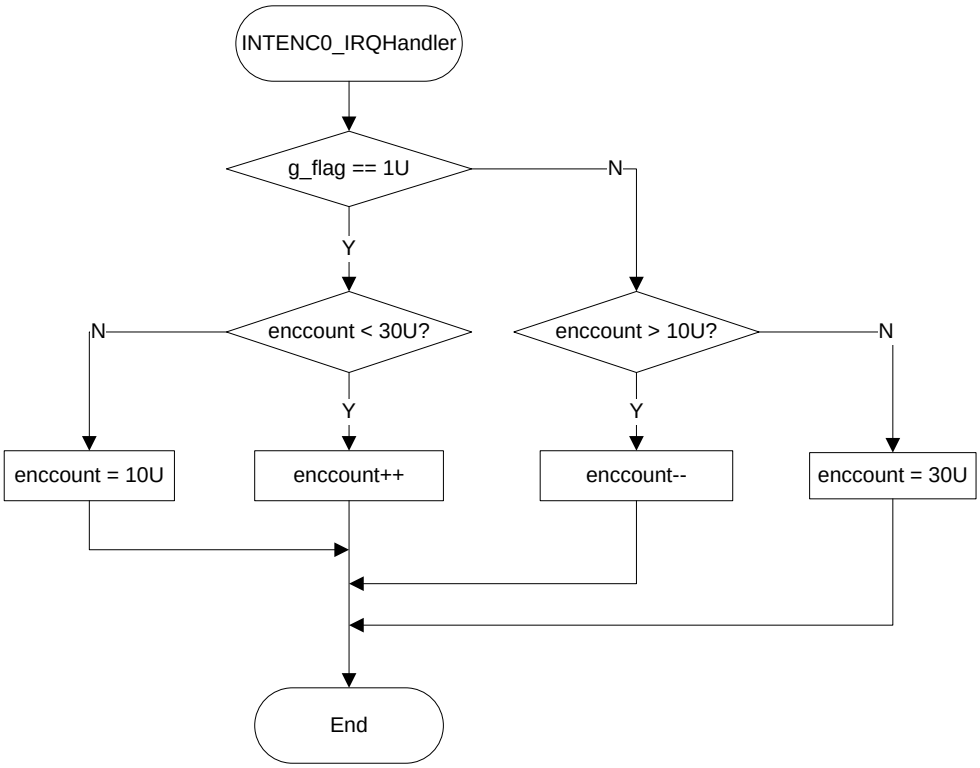
TX03 ペリフェラルドライバ(ENC, GPIO)を用いたサンプルプログラムです。

以下の例が含まれます:

1. ENC0 の初期化
2. ホイールマウスの回転検出

- **フローチャート**





- **サンプルプログラムのコードと説明**

TX03 ペリフェラルドライバ(ENC)を用いてホイールマウスの回転を検出するサンプルプログラムです。

まず、評価ボード上の LED を初期化します。

```
    /* LED initialization */
    LEDInit();
```

ENC 初期化構造体を用意し、ENC0 を初期化します。

```
    /* ENC0 initialization */
    m_enc.ModeType = ENC_ENCODER_MODE;
    m_enc.PhaseType = ENC_TWO_PHASE;
    m_enc.CompareStatus = ENC_COMPARE_DISABLE;
    m_enc.ZphaseStatus = ENC_ZPHASE_DISABLE;
    m_enc.FilterValue = ENC_FILTER_VALUE31;
    m_enc.IntEn = ENC_INTERRUPT_ENABLE;
    m_enc.PulseDivFactor = ENC_PULSE_DIV1;

    ENC_Init(EN0,&m_enc);
    ENC_SetCounterReload(EN0,0xFFFU);
```

GPIO を ENC0 に設定します。PD0 と PD1 は ENC0 入力に設定します。

```
    /* GPIO initialization */
    GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_0, ENABLE);/*Set PD0 as
input */
    GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_0);/*Set
PD0 as ENCA0 */
    GPIO_SetInputEnableReg(GPIO_PD, GPIO_BIT_1, ENABLE);/*Set PD1 as
```

```
input */
    GPIO_EnableFuncReg(GPIO_PD, GPIO_FUNC_REG_1, GPIO_BIT_1);/*Set
PD1 as ENCB0 */
```

ENC0 と ENC0 割り込みを許可します。

```
    /* Enable ENC0 */
    ENC_Enable(EN0);
    /* Enable ENC0 interrupt */
    NVIC_EnableIRQ(INTENC0_IRQn);
```

ホイールマウスの回転を検出するため ENC0 の回転方向検出状態を取得します。回転数に応じて LED1 を点滅します。

```
    /* LED1 blink speed based on the rolling of mouse wheel */
    while(1){
        f_enc = ENC_GetENCFlag(EN0);
        g_flag = f_enc.Bit.RotationDirection;
        LED_On(LED1);
        Delay2(enccount);
        LED_Off(LED1);
        Delay2(enccount);
    }
```

ENC カウントは、回転した数で、割り込みルーチンで変化します。
ホイールマウスが前方向に回転すると、ENC カウンタが 10 まで減ると ENC カウンタは 30 に戻り、ENC カウンタが 30 まで増えると ENC カウンタが 10 に戻ります。

```
    if(g_flag == 1U){
        if(enccount < 30U){
            enccount++;
        } else {
            enccount = 10U;
        }
    } else {
        if(enccount > 10U){
            enccount--;
        } else {
            enccount = 30U;
        }
    }
```

7-17 PMD

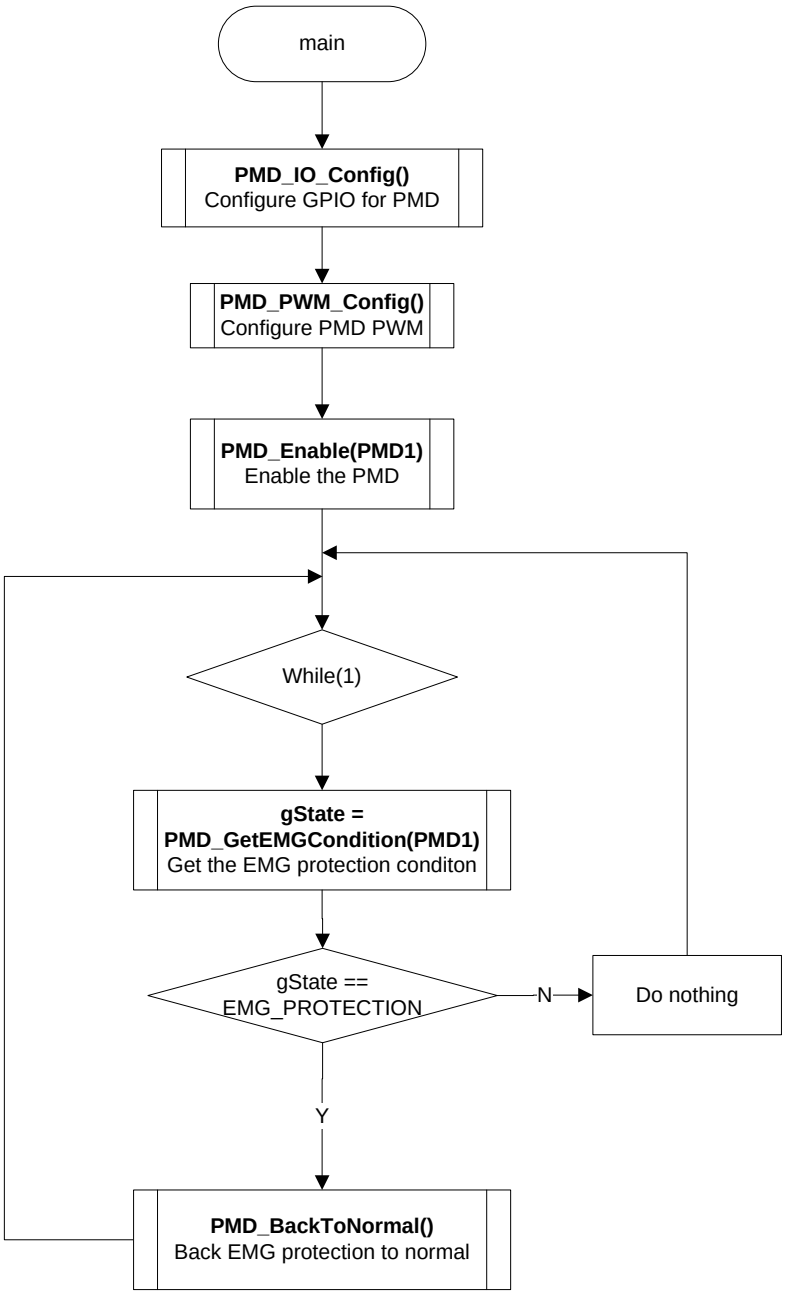
7-17-1 例: 位相出力

TX03 ペリフェラルドライバ (PMD, GPIO) のプログラムサンプルです。

この例では以下を行います。

1. PMD と GPIO の初期化
2. EMG 保護検出出力
3. EMG 保護検出出力から通常出力への状態変化

- **フローチャート**



- **サンプルプログラムのコードと説明**

まず LED、PMD、GPIO の初期化を行います。

```
/* LED initialization */
LEDInit();

/* GPIO configuration*/
PMD_IO_Config();

/* PMD PWM configuration*/
PMD_PWM_Config();
```

DEMO_U_PHASE 定義を行う場合、DUTY モードは U 相共通です。
DEMO_3_PHASE 定義を行う場合、DUTY モードは 3 相独立です。

```
/* PMD1 initialization */
    m_pmd.CycleMode = PMD_PWM_NORMAL_CYCLE;
#ifdef DEMO_U_PHASE
    m_pmd.DutyMode = PMD_DUTY_MODE_U_PHASE;          /* U-phase in
common */
#endif
#ifdef DEMO_3_PHASE
    m_pmd.DutyMode = PMD_DUTY_MODE_3_PHASE;          /* 3-phase
independent */
#endif
    m_pmd.IntTiming = PMD_PWM_INT_TIMING_MINIMUM;
    m_pmd.IntCycle = PMD_PWM_INT_CYCLE_1;
    m_pmd.CarrierMode = PMD_CARRIER_WAVE_MODE_1;     /* PWM mode 1
(center PWM, triangle wave) */
    m_pmd.CycleTiming = 0x3FFFU;

    PMD_Init(PMD1, &m_pmd);
```

3 相のコンペア値を設定します。
DUTY モードが U 相共通の場合、U 相と同じ出力です。
DUTY モードが 3 相独立の場合、Y/V/W それぞれの出力です。

```
    PMD_SetAllPhaseCompareValue(PMD1,0x0FFFU,0x1FFFU,0x2FFFU);
```

PMD 動作を許可します。

```
    PMD_Enable(PMD1);
```

EMG 保護検出の判定を行います。保護検出状態の場合は LED1 点灯し、EMG 保護検出出力から通常出力へ戻します。保護検出状態ではない場合は LED1 消灯します。

```
    while(1){
        /* Get the EMG protection condition*/
        gState = PMD_GetEMGCondition(PMD1);
        /* Judge the EMG protection condition */
        if (gState == EMG_PROTECTION) {
            /* LED1 will light on if the condition is in protection */
            LED_On(LED1);
            /* Back EMG protection to normal */
            PMD_BackToNormal();
            Delay(1000U);
        } else {
            /* LED1 will light off if the condition is not in protection */
            LED_Off(LED1);
        }
    }
```